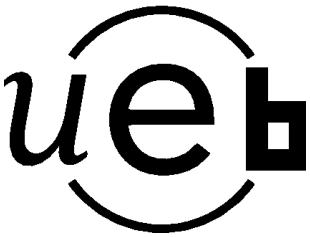
Classification et caractérisation de familles enzymatiques à l'aide de méthodes formelles

Gaëlle Garet

N^o d'ordre : 00000 ANNÉE 2015



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

École doctorale Matisse

présentée par

Gaëlle GARET

préparée à l'unité de recherche Inria/Irisa – UMR6074
Institut de Recherche en Informatique et Système Aléatoires
Composante universitaire : ISTIC

Classification et caractérisation de familles enzymatiques à l'aide de méthodes formelles

Thèse à soutenir à Rennes
le 16 décembre 2014

devant le jury composé de :

Jean-Christophe JANODET
Professeur à l'Université d'Evry-Val-d'Essonne / *Rapporteur*
Amedeo NAPOLI
Directeur de recherche au Loria, Nancy / *Rapporteur*
Colin DE LA HIGUERA
Professeur à l'Université de Nantes / *Examinateur*
Olivier RIDOUX
Professeur à l'Université de Rennes 1 / *Examinateur*
Mirjam CZJZEK
Directrice de recherche CNRS, Roscoff / *Examinatrice*
Jacques NICOLAS
Directeur de recherche à Inria, Rennes / *Directeur de thèse*
François COSTE
Chargé de recherche à Inria, Rennes / *Co-directeur de thèse*

Ainsi en était-il depuis toujours. Plus les hommes accumulaient des connaissances, plus ils prenaient la mesure de leur ignorance.
Dan Brown, Le Symbole perdu

Tu me dis, j'oublie. Tu m'enseignes, je me souviens. Tu m'impliques, j'apprends.
Benjamin Franklin

Remerciements

Je remercie tout d'abord la région Bretagne et Inria qui ont permis de financer ce projet de thèse.

Merci à Jean-Christophe Janodet et Amedeo Napoli qui ont accepté de rapporter cette thèse et à Olivier Ridoux, Colin De La Higuera et Mirjam Czjzek pour leur participation au jury.

J'aimerais aussi dire un grand merci à mes deux directeurs de thèse : Jacques Nicolas et François Coste, qui m'ont toujours apporté leur soutien tant dans le domaine scientifique que personnel. Jacques, merci pour ta patience, ta gentillesse et ton soutien à toute épreuve. Un grand merci surtout pour tout ce que tu m'as apporté au niveau scientifique, entre autre pour formaliser les problèmes rencontrés, et pour ta pile de bibliographie amassée pendant des années :) François, merci de m'avoir permis de faire mon stage de master qui m'a conduite ici, merci pour les discussions, constructives ou non, qui m'ont aidé à mieux comprendre les grammaires et les concepts qu'elles impliquent. Merci aussi de m'avoir donné l'occasion de découvrir le monde durant ces trois ans aux travers des voyages, je me souviendrais longtemps des voyages, Washington à l'occasion d'ICGI, Cordoba, les visites, le parc des condors ;)

Merci à tous ceux qui ont collaboré de près ou de loin à mes travaux, je pense notamment aux biologistes de la station de Roscoff qui m'ont toujours accueillie avec beaucoup de patience (et il en faut pour expliquer les concepts biologiques à une informaticienne !). En vrac, Myrjam, Gurvan, Thierry, Agnès, Catherine et tous ceux que j'oublie sur le moment.
Merci aussi à ceux qui m'ont accueillie dans leur labo en Argentine ou en Allemagne lors de mes différents séjours.

Un grand merci au groupe Symbiose (Dyliss, Genscale et Genouest), surtout ne changez rien ! ! Malgré les multiples départs et les arrivées, l'ambiance est toujours restée la même : excellente, ces 3 ans n'auraient pas été les mêmes sans chacun d'entre vous. Je me rappellerai des séminaires au vert, des midi activités, notamment les tournois de ping pong, l'escalade, le badminton (si si des fois j'y allais), les jeudi soirs amarillys, les afters de folie ou les samedi matins au marché, le voyage en Roumanie et j'en passe ... Tellement de bons souvenirs !
Merci à Mathilde de m'avoir supportée dans le bureau pendant ces 3 ans (enfin plutôt 2 ;)), les tours de smarties, les aprems entre filles, ...
Merci à ceux qui m'ont permi de squatter leur bureau sur la fin (Julie, Charles et Ayme-

rick) et à Claudia qui m'a supportée pendant la fin de la rédaction. Merci à Catherine, Coline ou encore Anaïs pour les pauses potins quand la rédaction devenait difficile. Un merci particulier à ceux qui ont toujours été là à n'importe quelle heure du jour ou de la nuit, Mathilde, Vincent, Sylvain : le stage de M2, les discussions mc do, les conférences, les séjours à Roscoff, ...

Merci à tous ceux qui m'ont fait découvrir la vulgarisation scientifique et qui nous ont aidé dans l'organisation de Sciences en Cour[t]s, une aventure à la fois scientifique et humaine qui a permis de penser à autre chose de temps en temps. Notamment les orgas, parmi eux Sylvain, Coraline, Charles, Marie, Kévin, Nico, ... mais aussi tous les réals ! Et je souhaite bon courage aux nouveaux (entre autre Clovis, Fanny et Paulin) qui ont accepté de reprendre le flambeau pour continuer l'aventure !

Merci à mes amis de Quimper qui m'ont écoutée pendant un certain nombre d'heure parler de cette thèse sans rien comprendre 'Je pense notamment à Camille, Elise, Céline, JR, Maureen, Kane, ... Merci à vous d'être là depuis nos années déjantées du lycée.

Et merci à toute ma famille qui m'a soutenue et encouragée dans cette aventure qu'est la thèse (même quand je n'étais pas toujours facile à vivre !), maman, Nico, Pierre, Audrey, Caro, ...

Enfin, merci à tous ceux que je n'ai pas cités mais qui se reconnaîtront.

Table des matières

Introduction

Prédire l'activité d'une enzyme à partir de sa séquence en acides aminés est une tâche d'une importance majeure pour la compréhension des réactions biochimiques avec de nombreuses retombées dans le domaine des biotechnologies. Cette thèse considère le problème de l'apprentissage de signatures caractéristiques de familles d'enzymes. Les enzymes sont des protéines particulières et on dispose déjà d'un certain nombre d'outils pour caractériser des familles de protéines et découvrir de nouveaux membres à ces familles. Le problème général est celui de l'inférence de modèles à partir d'un ensemble de séquences partageant une fonction commune. Parmi les modèles les plus utilisés, on trouve les modèles de Markov cachés ou encore les langages réguliers. Dans cette thèse nous nous intéressons à des modèles algébriques plus expressifs qui sont capables de capturer des interactions entre éléments éloignés sur la séquence de la protéine.

Dans ce but, nous nous sommes intéressés dans un premier temps à l'inférence de grammaires hors-contexte. Il s'agit d'un problème pour lequel il existe un certain nombre d'algorithmes heuristiques pour le traitement automatique des langues. Tous sont basés sur le principe de substituabilité de Harris pour lequel le travail de A. Clark [CE07] offre un cadre théorique et des résultats d'apprenabilité en introduisant la classe des langages hors-contexte substituables. Nous avons étendu ce principe en introduisant de nouvelles classes de langages et de nouveaux critères de généralisation basés sur des classes de substituabilité locales et/ou contextuelles. Ces nouveaux critères permettent ainsi de traiter des ensembles de séquences de protéines pour lesquels les exemples sont moins nombreux et les séquences beaucoup plus longues que pour les langues naturelles.

Afin de pouvoir utiliser ce modèle en pratique sur des ensembles de données réels, nous proposons un algorithme d'apprentissage efficace permettant de réduire la grammaire tout au long de l'apprentissage jusqu'à l'obtention d'une grammaire canonique non redondante du langage substituable cible. Nous avons ainsi obtenu de bons résultats sur des données réelles avec une haute spécificité et une bonne sensibilité grâce à une généralisation basée sur la substituabilité locale.

En pratique, la disponibilité des familles d'enzymes n'est pas immédiate. Il existe des groupes de séquences phylogénétiquement liées, appelés superfamilles, qui possèdent des motifs communs pour les repérer. Au sein de ces superfamilles, il existe un certain nombre de familles avec des activités diverses déterminées expérimentalement. Le coût et la difficulté des expérimentations ne permettent pas actuellement d'avoir des séquences d'apprentissage pour chaque famille existante.

Nous avons donc abordé le problème de la prédiction de l'appartenance d'une séquence à une famille. Nous présentons un classifieur basé sur l'identification de blocs de sous-séquences communes entre des séquences de familles connues et inconnues appartenant à la même superfamille. Nous nous appuyons pour cela sur la recherche de concepts formels construits sur le produit des blocs et des séquences. Contrairement à la plupart des classifieurs dans ce domaine, nous avons introduit les classes (familles) comme des objets à part entière. Nous traitons le problème non supervisé de la détection de nouvelles familles dans les séquences non étiquetées comme un problème d'optimisation en minimisant le

nombre de nouvelles familles tout en maximisant le support d'une nouvelle famille en terme de blocs caractéristiques. Nous avons utilisé ce classifieur sur des données provenant du génome d'une algue brune récemment séquencée, *Ectocarpus siliculosus*, relativement éloignée des espèces habituellement étudiées.

La dernière étape de l'étude est la création de grammaires hors-contexte pour chaque famille repérée au sein d'une superfamille. Ces caractérisations expressives permettent potentiellement de repérer des interactions intéressantes au sein des protéines.

Dans un premier temps, le chapitre 1 est consacré à l'introduction des concepts biologiques nécessaires à la compréhension de cette thèse. Il présentera les notions d'enzymes et de familles de séquences, ainsi que certaines méthodes utilisées pour caractériser un ensemble de séquences protéiques. Nous verrons ainsi les limites des approches utilisées actuellement.

puis, le chapitre 2 introduit les concepts mathématiques et informatiques utilisés pour modéliser les familles d'enzymes. Nous présentons dans ce but la théorie des langages et l'inférence grammaticale, et insistons sur l'état de l'art utilisant des langages réguliers.

Dans le chapitre 3, nous cherchons à obtenir des grammaires de la classe des grammaires hors-contexte, plus expressives que les grammaires régulières classiquement utilisées. Dans ce but, nous introduisons les concepts de substituabilités locale et contextuelle comme critères sur lesquels s'appuiera la généralisation des séquences d'apprentissage.

Le chapitre 4 est consacré à l'apprentissage supervisé et non supervisé des propriétés des différentes sous-familles présentes dans l'échantillon d'apprentissage grâce à l'analyse de concepts formels. Il sera ainsi possible de discriminer des ensembles de séquences présentant des activités inconnues. Cela permettra d'introduire une phase de sélection dans la méthode présentée au chapitre 2.

Enfin, le chapitre 5 présentera les perspectives et conclusions de ce travail.

Chapitre 1

Modélisation de familles d'enzymes

Cette thèse étudie le problème de la reconnaissance de familles d'enzymes par des méthodes formelles. Il s'agit d'un problème important en bioinformatique à la fois d'un point de vue applicatif (les enzymes jouent un rôle dans la plupart des réactions biologiques et ont des implications en biotechnologie) et fondamental (comment caractériser des ensembles de séquences ?).

Nous commençons par introduire le problème biologique et ses différentes problématiques, puis nous présentons les limites des représentations existantes.

1.1 Problème biologique : reconnaissance de classes d'enzymes

1.1.1 Un peu d'histoire

Depuis la découverte de l'ADN par Watson et Crick en 1953 [WC+53] et la première technique de séquençage développée en 1975 [SC75], l'Homme a cherché à décrypter les informations contenues dans cet ADN pour mieux comprendre le fonctionnement de la vie [GK10]. De grands projets de séquençage de génomes, notamment celui de l'espèce humaine, ont alors vu le jour et les premières bases de données de connaissances génomiques cherchant à répertorier les connaissances acquises sont apparues.

C'est également dans les années 1970 que l'on voit apparaître le terme de *bioinformatique*. La bioinformatique peut être vue comme l'ensemble des concepts et des techniques nécessaires à l'interprétation informatique de l'information biologique. En effet, les données générées par le séquençage ne peuvent pas toutes être interprétées expérimentalement, et il devient nécessaire de trouver des techniques automatisées pour les annoter.

Alors que pour obtenir une esquisse du génome humain comprenant une séquence de 3 milliards de nucléotides, il a fallu plus de 11 ans de travail dans des centaines de laboratoires à travers le monde et 3 milliards de dollars [L+91], aujourd'hui, grâce aux connaissances accumulées et à de nouvelles techniques de séquençage, il est possible de

séquencer un génome en quelques jours, voire quelques heures seulement pour quelques milliers d'euros.

Cette avancée technologique a donné lieu à une explosion de données et à de nouveaux enjeux au niveau informatique. Il faut notamment trouver de nouvelles méthodes pour stocker, trier, répertorier, annoter de telles données et automatiser ces processus. [ZCBZ11] apporte une vue d'ensemble sur ces problématiques liées à l'apparition du séquençage haut-débit.

L'annotation de séquences, qui va donner un sens à toute cette information, ne peut plus se pratiquer de manière manuelle. Même la vérification des annotations automatiques (curation) devient difficile manuellement et il est indispensable de mettre en place des algorithmes d'apprentissage de plus en plus sophistiqués permettant de tirer parti au mieux des données déjà annotées afin de traiter celles qui ne le sont pas encore. On trouvera dans le livre [BB01] une très bonne introduction aux problèmes d'apprentissage en bioinformatique. Celui qui nous intéresse ici, c'est à dire l'annotation fonctionnelle de protéines à partir de leurs séquences y est décrit comme un problème difficile. Nous nous intéressons ici particulièrement aux protéines possédant une fonction catalytique : les *enzymes*, dont la première a été caractérisée en 1933 [PP33].

1.1.2 Qu'est-ce qu'une enzyme ?

Dans ce manuscrit, nous nous intéressons à un ensemble de macro-molécules essentielles pour les cellules vivantes : les *enzymes*. Nous commençons donc par présenter ces objets d'étude. Une enzyme est une protéine qui joue un rôle de catalyseur biologique (ou biocatalyseur), c'est-à-dire de composé qui facilite une réaction biochimique sans en modifier les produits.

Du point de vue informatique, nous considérerons une enzyme comme un triplet, de la même manière que pour toute autre protéine. En effet, une protéine peut être considérée à plusieurs niveaux :

1. une séquence d'acides aminés (structure primaire) ;
2. une structure formée par cette séquence repliée dans l'espace (structure spatiale) ;
3. une fonction associée.

Les protéines résultent de la transcription d'une séquence d'ADN en séquence d'ARN puis de la traduction de cet ARN en séquence d'acides aminés qui constitue la protéine. Les protéines sont des éléments essentiels de la vie de la cellule : elles peuvent jouer un rôle structurel, un rôle dans la mobilité, un rôle catalytique, un rôle de régulation de la compaction de l'ADN ou d'expression des gènes, etc. En somme, l'immense majorité des fonctions cellulaires est assurée par des protéines.

Chimiquement, il s'agit d'une macromolécule composée d'une ou plusieurs chaînes d'acides aminés liés entre eux par des liaisons peptidiques (chaîne polypeptidique). L'ordre dans lequel les acides aminés s'enchaînent est codé dans le génome et constitue la structure primaire de la protéine. La protéine se replie sur elle-même pour former des structures, dont les plus importantes quantitativement sont l'hélice alpha et le feuillet bêta. Elles sont

stabilisées par des liaisons hydrogènes entre les atomes de carbone et d'azote ce qui introduit alors des corrélations, courte et longue distance, entre les chaînes d'acides aminés et forme la structure 3D appelée structure tertiaire.

L'enzyme se distingue des autres protéines par sa fonction de catalyseur.

<table>
<tr><th colspan="2">exemple : la β-agarase [AJH+03]</th></tr>
<tr><td>Séquence</td><td>MDIAQDWNGIPVPANPGNGMTWQLQDNVSDSFNYTSSEGNRPTAFTSKWKPSYINGWTGPGS
TIFNAPQAWTNGSQLAIQAQPAGNGKSYNGIITSKNKIQYPVYMEIKAKIMDQVLANAFWTL
TDDETQEIDIMEGYGSDRGGTWFAQRMHLSHHTFIRNPFTDYQPMGDATWYYNGGTPWRSAY
HRYGCYWKDPFTLEYYIDGVKVRTVTRAEIDPNNHLGGTGLNQATNIIIDCENQTDWRPAAT
QEELADDSKNIFWVDWIRVYKPVAVSGGGNNSLEHHHHHH
X : résidu catalytique, x : site actif</td></tr>
<tr><td>Structure</td><td>β jelly-roll
  </td></tr>
<tr><td>Fonction</td><td>agarase
elle permet à certaines bactéries de dégrader l'agar comme source de carbone
La réaction qu'elle catalyse dans la bactérie Zobellia galactanovorans
est la suivante :
$agorose + H2O \longrightarrow neoagarotetraose$</td></tr>
</table>

La fonction d'une enzyme est définie par la réaction chimique qu'elle facilite. Chaque enzyme est associée à un *substrat*, molécule sur laquelle elle agit. L'interaction est possible grâce à la forme (structure 3D) de l'enzyme qui permettra au substrat et à l'enzyme de s'emboîter comme les deux pièces d'un puzzle afin de former un complexe transitoire enzyme-substrat. Ce mécanisme initialement décrit dans [KJ58] a fait l'objet de nombreuses études depuis et nous présentons brièvement la vision actuelle issue de [SH08]. L'enzyme ne prend sa forme active qu'en présence de son substrat. La zone d'accroche au niveau de l'enzyme est appelée *site actif*. En général, il n'y a que quelques acides aminés de la séquence de l'enzyme qui interagissent avec le substrat, ils sont appelés *résidus catalytiques*. Une fois ce complexe formé, l'enzyme peut agir sur son substrat, par exemple permettre sa dissociation en deux composés. La forme particulière d'une enzyme est définie en grande partie par sa séquence et les différentes propriétés physico-chimiques des acides aminés la composant (hydrophobicité, charge, ...).

Un des problèmes majeurs est de réussir à prédire la fonction d'une enzyme (type de réaction catalysée et substrat) grâce à sa séquence d'acides aminés. Malgré le dévelop-

pement d'un grand nombre de méthodes, cela reste encore un problème difficile de par la grande variabilité des mécanismes impliqués [DV00, RCO+13].

1.1.3 Des familles de séquences

L'un des principes permettant l'annotation fonctionnelle de séquences de protéines repose sur le fait que, dans la grande majorité des cas, deux séquences similaires avec un fort pourcentage d'identité de séquences auront probablement la même structure et partageront alors la même fonction [CL86b, WP99]. En effet, on suppose que les mutations gardées au fil du temps sur une séquence tendent à conserver sa structure et sa fonction. On parle alors de séquences homologues. Celles-ci dérivent d'un ancêtre commun et partagent des séquences, des structures et fonctions similaires . Les familles de protéines sont alors souvent définies comme un ensemble de séquences homologues.

Il existe de nombreuses bases de données qui tirent partie de l'homologie de séquences, à partir de la présence de certains motifs ou domaines, pour classifier les séquences de protéines [C+14, HJM+12, JHH96].

Une famille ainsi définie par similarité de séquence possède-elle une unique fonction ? Si l'on est capable de définir la famille à laquelle appartient une nouvelle séquence, pourra-t-on lui assigner la même fonction que ses parents ? Ce n'est malheureusement pas si simple ...

Un grand nombre d'articles traite du lien existant entre séquence-structure-fonction [KG02, BTB00, Gu03, FW03, WKG00] et les résultats de multiples expériences montrent, même s'il ne s'agit pas de la majorité des cas, que certaines séquences tendent à diverger fonctionnellement avec le temps. En effet, au fil des mutations et adaptations, les protéines se sont spécialisées pour une meilleure interaction avec leur environnement.

Il peut ainsi être difficile de déterminer les limites de chaque famille possédant une même fonction, et ce d'autant plus que le pourcentage d'identité à considérer diffère pour chaque famille. Deux enzymes possédant 40% d'identité peuvent posséder des fonctions différentes alors que d'autres avec 25% d'identité peuvent partager la même fonction. Un exemple d'arbre phylogénique se trouve en figure 1.1 qui montrent que certaines séquences très proches ne possèdent pas la même fonction.

Pour palier ces difficultés, les protéines peuvent être regroupées en familles à différents niveaux : séquentiel comme un groupe de séquences homologues mais aussi structural et fonctionnel.

Il existe ainsi des bases de données de familles structurales, définies hiérarchiquement ([AV12] [FBC14], [OMJ+97]) selon la structure des protéines, le nombre et la forme des structures secondaires, la forme du repliement ... Il est à noter que dans ces familles, les membres peuvent ne pas être similaires en terme de séquences. Ainsi une même structure peut être partagée par deux séquences très différentes non homologues. Une structure/fonction peut en effet être recréée plusieurs fois indépendamment au cours du temps. Il s'agit d'un mécanisme de convergence évolutive (GH16 et GH12).

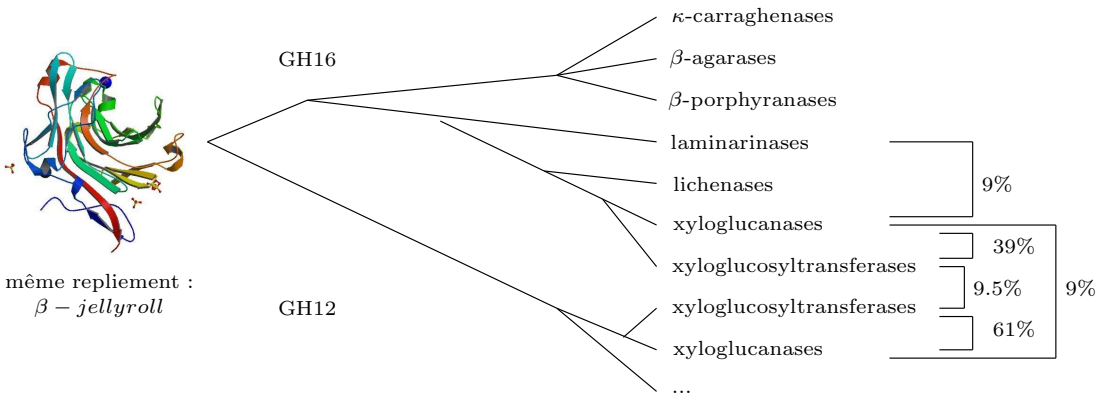


Fig. 1.1 – Exemple d'arbre phylogénétique sur la superfamille des Glycosides Hydrolases. Sur la droite on trouve les différentes fonctions qui peuvent être rencontrées dans cette superfamille avec les pourcentages d'identité de séquences retrouvées entre différentes familles fonctionnelles. Par exemple, on trouve 60% d'identité de séquence entre deux séquences de fonction différentes (xyloglucanase et xyloglucosyltranférase dérivant des GH12)
alors qu'il n'y a que 9% d'identité entre deux séquences de même fonctions mais dérivant de deux branches évolutives différentes.

Enfin, on peut définir des familles de protéines fonctionnelles. Il existe différentes hiérarchies pour définir ce type de familles toutes basées sur différents critères. La hiérarchie la plus utilisée pour caractériser les enzymes est celle de la classification EC, sur laquelle nous reviendrons un peu plus loin, qui définit le type de réaction et le substrat de l'enzyme. Mais il existe d'autres classifications fonctionnelles concernant les protéines dans leur globalité basées par exemple sur la localisation de la protéine dans la cellule, son action, ...[MHK+99, RZM+04, SCD+13, EPS+05]. Encore une fois, certaines enzymes montrent des fonctions similaires sans pour autant être homologues.

Une des difficultés majeures des biologistes confrontés à une nouvelle séquence enzymatique est d'identifier sa fonction, c'est pourquoi nous avons choisi de traiter le problème de la modélisation et la reconnaissance de familles fonctionnelles. Pour se rendre compte de l'ampleur du problème, la table 1.1 fournit quelques chiffres concernant l'annotation des séquences protéiques de la base UniprotKB [C+14] qui possède deux bases distinctes, l'une qui annote les nouvelles séquences de manière totalement automatique, sans intervention humaine et l'autre où les annotations sont vérifiées par des biologistes experts. On remarque que les séquences prédites expérimentalement sont bien moins nombreuses que celles annotées par similarité, même dans une base vérifiée manuellement, ce qui pose des problèmes pour repérer des divergences de fonctions ou de nouvelles fonctions non encore caractérisées [DV00].

		TrEMBL (automatique)	SwissProt (manuel)
Nombre de séquences		83955074	546439
Annotation	expérimentale	1,24%	27%
	par homologie	23,69%	70,6%
	prédite	75,07%	2,1%
	incertaine	0%	0.3%

Tab. 1.1 – L'annotation fonctionnelle des protéines en quelques chiffres (1^{er} octobre 2014)

Un aspect essentiel de notre travail sera d'utiliser des modèles expressifs afin de pouvoir déterminer les séquences de différentes fonctions au sein d'une famille phylogénétiquement liée.

Dans la suite, nous utiliserons les définitions de [GB00] qui évoque ce problème. Un ensemble de séquences homologues possédant les mêmes mécanismes enzymatiques (mêmes résidus catalytiques et même fonctionnement) mais différentes fonctions (en particulier différents substrats) est appelée une *superfamille mécaniquement divergente*, que nous abrégeons dans la suite en *superfamille*. Un ensemble de séquences homologues avec une même fonction est appelé une *famille*.

1.1.4 Le problème de l'annotation d'enzymes à partir de séquences

Afin de répertorier les différentes réactions pouvant être catalysées par une enzyme, la classification la plus utilisée est la classification EC. A l'origine, elle a été mise en place en 1955 par l'IUBMB (International Union of Biochemistry and Molecular Biology), EC est le sigle de *Enzyme Commission*.

C'est une classification numérique des enzymes, basée sur la réaction chimique qu'elles catalysent. En tant que système de nomenclature des enzymes, chaque numéro EC est associé à un nom recommandé pour l'enzyme correspondante.

Un numéro EC est de la forme $A.B.C.D$ où A,B,C et D sont des entiers. Ces nombres représentent chacun une étape dans la précision de la classification de l'enzyme. Les quatre nombres de la nomenclature EC des enzymes désignent chacun une caractéristique de l'enzyme qui permet de l'identifier. Le premier nombre de la nomenclature EC indique le type de réaction catalysée, le second le substrat général impliqué lors de la réaction, le troisième le substrat spécifique impliqué et le quatrième le numéro de série de l'enzyme. Le niveau de généralité des différents niveaux peut varier suivant le type de réaction. Par exemple, l'enzyme β-agarase a le code EC 3.2.1.81 qui est construit comme suit : 3 dénote une hydrolase qui catalyse la réaction $R-R'+H2O \leftrightharpoons R-OH+R'-H$, où $R-R'$, R et R' dénotent des molécules. L'enzyme permet de rompre la liaison existante entre R et R' à l'aide d'une molécule d'eau. Le groupe 3.2 regroupe les glycosylases, hydrolases agissant sur des liaisons glycosidiques (R se termine alors par ce type de liaison dans l'équation ci-dessus). 3.2.1 indique plus précisément les glycosidases, enzymes qui agissent sur des

	EC 1	EC2	EC3	EC4	EC 5	EC 6	Total
Nombre de sous-classes	1550	1644	1302	580	251	182	5509
Nombre de séquences annotées	12127	28549	17198	7940	4837	9393	80044

Tab. 1.2 – Répartition des classes et nombres de séquences d'enzymes selon le premier niveau de la hiérarchie EC au 1^{er} octobre 2014

liaisons S-glycosyl ou O-glycosyl et 3.2.1.81 indique qu'elle permet de cliver une liaison spécifique au substrat de l'agar (qui possède une liaison 0-glycosyl).

Le niveau supérieur de cette classification est le suivant :

- EC 1 Oxydo-réductases : catalysent les réactions d'oxydo-réduction ;
- EC 2 Transférases : transfèrent un groupement fonctionnel (par exemple un groupe méthyle ou phosphate) ;
- EC 3 Hydrolases : catalysent l'hydrolyse de diverses liaisons ;
- EC 4 Lyases : brisent diverses liaisons par d'autres procédés que l'hydrolyse et l'oxydation ;
- EC 5 Isomérases : catalysent les réactions d'isomérisation dans une simple molécule ;
- EC 6 Ligases : joignent deux molécules par des liaisons covalentes.

La nomenclature complète est disponible à l'adresse `http://www.chem.qmul.ac.uk/iubmb/enzyme/`.

La classification est régulièrement mise à jour au fur et à mesure des découvertes [TB00], et même si elle possède quelques limites [Web64, Web93], comme pour les enzymes multi-fonctionnelles, ou une hiérarchie trop peu profonde pour certaines fonctions, elle reste la plus utilisée à travers le monde.

Pour donner une idée du nombre de réactions connues et de l'évolution au fil des ans, la classification EC possédait à ses débuts, en 1962, 712 classes à 4 chiffres, en 1992, elle en possédait déjà 3196 et aujourd'hui en 2014, le nombre de classes connues est de 5509.

La répartition des différentes classes selon le premier niveau de la hiérarchie est montrée en table 1.2. On peut aussi constater que le nombre de séquences de SwissProt qui ont été annotées par un numéro EC, soit 80000 sur 546000, représente une grande partie des séquences protéiques (une sur sept).

Au sens strict du terme, la nomenclature EC ne spécifie pas d'enzymes, mais des réactions catalysées par des enzymes. Des enzymes provenant de différents organismes et catalysant la même réaction reçoivent le même code EC. Pour identifier de manière unique une protéine par sa séquence d'acides aminés, on peut utiliser les identifiants SwissProt et la base ENZYME [Bai00] (http ://www.expasy.org/enzyme) qui regroupe codes EC, identifiants SwissProt, liens vers d'autres bases et références bibliographiques.

Il existe d'autres bases de données possédant ces informations comme Macie [HAB+07] ou Brenda [SCP+13].

Il existe également un grand nombre d'outils permettant de scanner une séquence et qui donnent en résultat la classe présumée de cette séquence. Afin de pouvoir annoter une séquence avec un numéro EC, il est nécessaire de pouvoir la comparer avec les séquences de la base et de mettre au point un score qui permettra de discriminer la famille à laquelle elle appartient. Il existe plusieurs méthodes pour arriver à un tel résultat. L'une consiste à comparer la séquence avec toutes les séquences de la base et à sélectionner celle qui s'en rapproche le plus. La deuxième consiste à créer une base de signatures communes à un ensemble de séquences partageant une fonction et ensuite chercher ces signatures dans la séquence à classer [BHKM06]. Nous nous situons dans cette dernière approche.

Nous avons vu précédemment que les méthodes de recherche d'homologie de séquences seules risquent de générer des faux positifs lors de l'annotation, en particulier lors de la caractérisation d'une nouvelle fonction au sein d'une surperfamille d'enzymes où nous avons vu que seuls quelques acides aminés entraient en jeu lors de la catalyse. De plus, nous voulons essayer de caractériser une fonction par un modèle explicite afin de comprendre le fonctionnement d'une classe d'enzymes à partir de ses séquences. La section suivante discute des méthodes couramment employées d'apprentissage de modèles à partir de séquences protéiques.

1.2 Modélisation d'une famille à partir d'un ensemble de séquences

Nous nous intéressons ici aux méthodes les plus courantes pour modéliser un ensemble de séquences homologues appartenant à une même famille. Nous présenterons en particulier celles (dites "boites blanches") conduisant à un modèle explicite. En effet, nous souhaitons nous focaliser sur celles permettant aux biologistes de comprendre la décision de classer une séquence dans une certaine famille. Pour cela il est indispensable que le modèle soit compréhensible et lisible. Au vu des connaissances limitées dont dispose un système de classement automatique, celui-ci doit avoir pour objectif une aide à la décision argumentée plutôt qu'un rôle décisionnaire strict.

Dans la littérature, différents moyens existent pour détecter les caractéristiques des séquences appartenant à une même famille. Nous nous intéressons ici aux méthodes basées sur des techniques d'alignement des séquences.

1.2.1 Alignement de séquences

L'alignement de séquences est une manière de disposer en vis à vis les acides aminés de chaque séquence considérée pour maximiser les zones de concordance qui traduisent des similarités ou dissemblances. Un alignement est constitué d'un ensemble de séquences de même longueurs, obtenues en introduisant des blancs (gaps) dans le texte de chaque séquence d'origine, de façon à comparer les caractères position par position.

Les méthodes peuvent soit essayer d'aligner les séquences sur la totalité de leur longueur, on parle alors d'*alignement global*, soit se restreindre à des régions limitées dans lesquelles la similarité est forte, à l'exclusion du reste des séquences, on parle alors

d'*alignement local*. En effet, lorsque les séquences d'une même famille sont trop divergentes, il devient utile de s'intéresser uniquement aux fragments de séquence à forte similarité. Nous présentons d'abord ces méthodes pour l'alignement d'une paire de séquences.

– Alignement global (sur toute la longueur des séquences)

Les alignements globaux sont plus souvent utilisés quand les séquences considérées sont similaires et de tailles comparables. Une technique générale, proposée par de Needleman-Wunsch [NW70] par programmation dynamique, permet de réaliser des alignements optimaux en temps quadratique par rapport à la taille des séquences. Le score associé à chaque alignement est basé sur le nombre d'acides aminés communs à chaque position et prend également en compte la probabilité qu'un acide aminé à une position soit remplacé par un autre suivant ses propriétés physico-chimiques. Par exemple deux acides aminés ayant tous les deux une charge positive auront une probabilité plus forte d'avoir été substitués au cours de l'évolution sur deux séquences possédant une fonction commune.

Il existe plusieurs matrices permettant d'obtenir un score de similarité entre deux acides aminés donnés. Les plus utilisées sont les matrices de Dayhoff [Day73] basées sur des distances évolutives entre espèces et les matrices de Henikoff [HH92], appelées BLOSUM, basées sur le contenu en information des substitutions. Différents articles traitent des avantages et inconvénients d'utiliser l'une ou l'autre [Mou08].

– Alignement local Il arrive fréquemment que la région homologue soit limitée à une partie des séquences. C'est le cas lorsque deux protéines partagent un domaine homologue, associé à une fonction commune, mais que le reste de leurs séquences est dissemblable. On utilise alors une méthode d'alignement local, comme l'algorithme de [SW81] basé aussi sur la programmation dynamique. Le programme BLAST [AGM+90] est une méthode heuristique rapide permettant d'effectuer des recherches dans les bases de données à partir d'une séquence requête que le programme segmente en sous-séquences de longueur fixée. Chacune des sous-séquences est ensuite comparée aux séquences de la base. Les méthodes locales utilisent un calcul de score adapté qui évite de pénaliser les régions non-homologues et ne calculent le score que sur la région conservée.

Avec des séquences très voisines, les résultats obtenus par les méthodes d'alignement local ou global sont très proches. Pour cette raison, les méthodes d'alignement local, plus flexibles, sont plus souvent utilisées aujourd'hui.

Alignement multiple Pour découvrir des modèles de familles, il est nécessaire d'aligner l'ensemble des séquences connues d'une famille.

On parle d'alignement multiple quand il s'agit d'aligner plus de deux séquences. Celui-ci peut être global comme celui utilisé dans ClustalW [THG94]. Un exemple d'alignement obtenu par cette méthode est donné en figure 1.2, où l'on peut voir que les séquences partagent une similarité importante même si elles contiennent des mutations ponctuelles. ClustalW recherche l'alignement de façon progressive. Il commence par aligner globalement toutes les séquences deux à deux pour obtenir une matrice de similarité entre les

séquences. L'algorithme effectue ensuite une classification hiérarchique ascendante des séquences en se basant sur cette matrice. Il commence par aligner les séquences les plus similaires entre elles, puis suit l'ordre de branchement dans l'arbre pour aligner les différents ensembles de séquences, sur la base d'un algorithme glouton.

D'autres méthodes permettent de calculer un ensemble d'alignements locaux, prenant en compte tout ou partie des séquences considérées, comme le programme Dialign2 [Mor99]. Chaque bloc est lui-même fait d'un ensemble de facteurs de même taille provenant d'un sous-ensemble des séquences d'apprentissage (avec seulement un facteur par séquence impliqué dans un bloc). Un bloc aligne implicitement ses facteurs par leurs positions relatives, c'est à dire que chaque position d'un bloc correspond à des caractères alignés des séquences impliquées. Dialign2 cherche à définir l'ensemble optimal des blocs qui composent l'alignement, et le score qu'il utilise a pour effet de maximiser la couverture en termes de séquences de ces blocs.

Ce type d'alignement est *consistant* [AM01], c'est à dire que les différents blocs ne peuvent pas se chevaucher pour une séquence donnée et qu'il conserve l'ordre d'apparition des blocs le long des séquences. Il n'est pas *uniforme* dans le sens où certains blocs ne couvrent pas l'ensemble des séquences.

Dans les différentes méthodes, l'ensemble des blocs découverts est trié selon un ordre défini par un score prenant en compte la similarité et le nombre de séquences impliquées. Le choix des blocs composant l'alignement est alors fait de manière itérative dans l'ordre des blocs définis et de façon à conserver un alignement consistant. Ainsi, on commence par intégrer dans l'alignement final le bloc possédant le score le plus haut. Puis, si les fragments de séquences impliqués dans le deuxième bloc ne sont pas déjà couverts par le premier, il est lui aussi intégré, et ainsi de suite jusqu'à épuisement des blocs.

D'autres programmes, comme [LNL95] ou BlockMaker [HHAP95] ne cherchent pas à satisfaire la contrainte de consistance et obtiennent donc des alignements multiples locaux permettant aux blocs ne pas apparaître dans le même ordre sur l'ensemble des séquences.

Ces différents types d'alignements sont présentés et comparés plus en détail dans [DA+00].

Un dernier type d'alignement existant est celui d'*alignement multiple local et partiel* (PLMA) développé dans [Ker08] et que nous utilisons par la suite.

Comme Dialign2, un PLMA est formé par un ensemble de blocs consistants, non chevauchants et non croisés. Mais contrairement aux choix effectués dans Dialign2, un PLMA ne cherche pas à maximiser le nombre de séquences impliquées dans un bloc et se base sur un score privilégiant les blocs très similaires, prenant souvent en compte un petit nombre de séquences. Le score définissant l'ordre dans lequel les blocs sont choisis pour définir un alignement consistant tend alors à trouver un alignement partiel des séquences, privilégiant la détection de divergence au sein d'un ensemble de séquences. Un exemple est montré en figure 1.3.

1.2.2 Découverte de modèles réguliers à partir d'un alignement multiple

A partir de ces alignements obtenus depuis un ensemble de séquences, il est possible, automatiquement ou manuellement, d'en retirer des modèles de séquences.

Ces modèles représentant une famille de séquences ont deux fonctions :

- ils servent d'une part à pouvoir reconnaître/générer de nouvelles séquences appartenant à la famille qu'ils représentent ;
- et d'autre part, ils peuvent permettre de mieux comprendre le fonctionnement de cette famille en exhibant ses particularités au niveau séquentiel.

Il existe un grand nombre de méthodes permettant la découverte de modèles à partir d'un alignement de séquences.

Différents types de modèles Parmi les modèles largement utilisés, on peut citer les matrices de scores position spécifiques (PSSM) [GME87]. La construction de ces matrices est basée sur le calcul de la fréquence de chaque acide aminé à une position donnée d'un alignement multiple. Elles sont facilement représentables graphiquement sous la forme d'un logo [SS90] comme sur la figure 1.4.

Parmi les modèles les plus efficaces, les profils HMM, basés sur une version simplifiée des modèles de Markov cachés [Edd98]. Ils fournissent un schéma analogue à celui des PSSM mais permettent en plus de prendre en compte des probabilités d'insertion et délétion pour chaque position. Ce type de représentation est par exemple utilisé dans la base de données PFAM [BCD+04] ou pour représenter des familles d'enzymes dans PRIAM [CRCFK03]. Eficaz est un autre programme permettant de calculer des profils HMM à partir d"un ensemble de séquences S d'enzymes possédant le même numéro EC [AHS09]. L'avantage de cette méthode est qu'elle utilise également un ensemble d'apprentissage négatif représenté par les séquences homologues à S mais ayant un numéro EC différent. Elle procède alors à deux alignements multiples de ces deux jeux de données et inclut dans son HMM les positions alignées des séquences positives, en excluant les positions alignées également dans l'échantillon négatif (indiquant un alignement dû à l'homologie de séquences uniquement, et non discriminant pour la fonction de l'enzyme).

La limitation majeure de ces méthodes est qu'elles ne tiennent compte que des informations statistiques sur la fréquence d'apparition des acides aminés à certaines positions et n'ont pas la possibilité de représenter une famille comme un enchaînement d'acides aminés se structurant dans l'espace. Elles ne tiennent pas compte de corrélations qui pourraient exister entre les positions.

Du fait de la nature hiérarchique de la numérotation EC pour les enzymes, il existe un certain nombre de méthodes construisant à partir d'un ensemble d'attributs caractéristiques hiérarchiques comme des arbres de décision [SY09, dKK+97] ou des SVMs hiérarchiques [WYD10]. La plupart de ces méthodes utilisent des attributs comme la proportion en acides aminés, la longueur des séquences ou le pourcentage d'acides aminés hydrophobes. Elles utilisent généralement des informations pouvant être extraites facilement mais sans réel pouvoir explicatif ou discriminant. EzyPred [SC07], semble plus intéressant

dans son choix d'attributs, et utilise des vecteurs renseignant sur la présence/absence de certains domaines fonctionnels connus en se servant de la base PFAM.

Il existe un autre type de modèle de signatures de familles de protéines plus explicite consistant à décrire les familles par un ensemble d'expressions régulières.

Les signatures les plus utilisées sont sans doute les motifs de la base de données Prosite [HBB+06]. Par exemple, le motif Prosite représentant les GH16, dont une séquence a été présentée plus haut est : $E-[LIV]-D-[LIVF]-x(0,1)-E-x(2)-[GQ]-[KRNF]-x-[PSTA]$. Les différentes positions sont séparées par des $-$, les crochets permettent de définir plusieurs alternatives à une position et les gaps de m à n acides aminés sont représentés par l'expression $x(m,n)$.

Les motifs de la base Prosite sont en général mis au point manuellement à partir d'alignements multiples mais il existe des programmes pour les générer à partir de séquences [JHH96, BBB+09], basés sur des techniques de découverte de motifs.

[BJEG98] présente une vue d'ensemble de ce type de techniques et propose un schéma général permettant la découverte de modèles :

1. choix d'un espace d'hypothèses : c'est le type de modèle à découvrir (expression régulière,...) ;
2. définition d'une fonction de score : elle reflète l'adéquation entre le modèle appris et les séquences d'entrée ;
3. développement d'un algorithme qui, depuis les données d'entrée (séquences ou alignement) retourne l'hypothèse possédant le score le plus élevé.

Le type de signatures présentées jusqu'ici permet d'exprimer facilement la succession d'acides aminés mais aussi les notions de gaps, substitutions, délétions et insertions possibles au sein des différentes séquences de la famille représentée.

Cependant, il reste encore difficile à ce stade de caractériser des relations entre les acides aminés à plus ou moins longue distance. Pourtant, plusieurs références [GSSV94] ont démontré qu'une mutation d'un acide aminé au cours de l'évolution peut entraîner des mutations des acides aminés en contact au niveau de la structure 3D. Il semble donc judicieux d'utiliser des modèles capables de représenter ces relations dans les langages appris. Pour cela, on s'intéresse dans le chapitre suivant à l'inférence de modèles grammaticaux plus généraux.

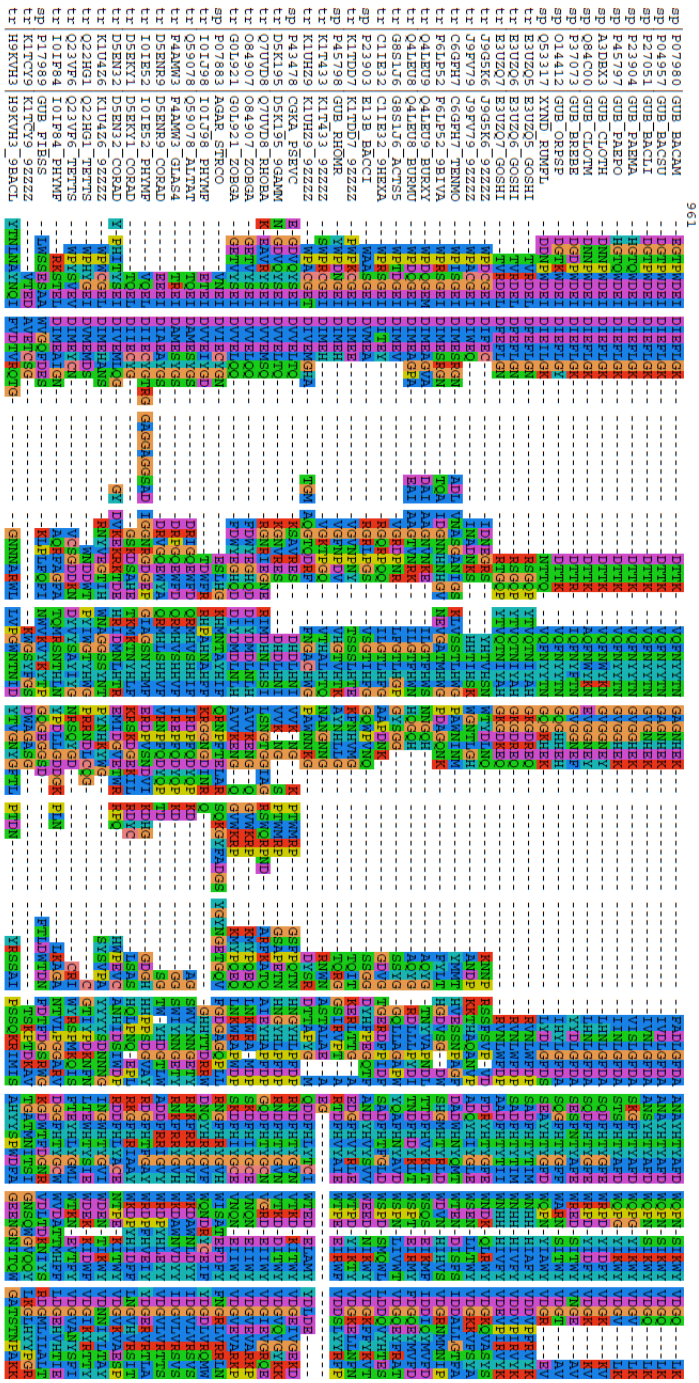


Fig. 1.2 – Alignement de séquences de protéines à l'aide de ClustalW. Chaque ligne représente une séquence d'acides aminés et chaque colonne représente une position alignée. Les différentes couleurs utilisées pour les acides aminés reflètent leurs propriétés physico-chimiques. Les tirets représentent les gaps insérés pour obtenir l'alignement.

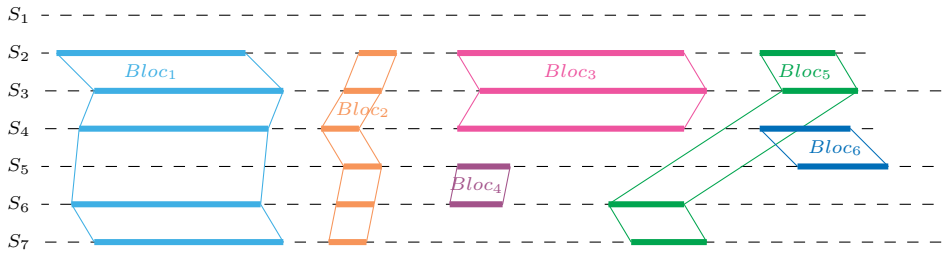


Fig. 1.3 – Représentation d'un alignement de séquences multiple local et partiel. Chaque ligne représente une séquence. L'alignement est composé de plusieurs blocs d'alignements locaux composés de fragments de séquences, chaque fragment impliqué dans un bloc est surligné, et chaque bloc est représenté ici par une couleur différente. Par exemple, le bloc 3 (représenté en rose ici) aligne des positions des séquences S_2, S_3 et S_4.

Fig. 1.4 – Logo obtenu après alignement multiple de séquences de la famille des GH16 et pondération des positions. Chaque position de l'alignement est représentée par une colonne. Sur chaque colonne, on trouve les acides aminés présents à cette position, la taille de l'acide aminé est en relation avec sa probabilité d'apparition à cette position. Ainsi, sur cet exemple, on remarque que certaines positions (où se trouvent seuls les acides aminés E et D sur une colonne) sont caractéristiques de cette famille.

Chapitre 2

Inférence grammaticale sur des séquences biologiques

Comme nous l'avons vu dans le chapitre précédent, la connaissance des séquences biologiques (ADN, ARN, Protéine) permet de déterminer l'ensemble des réactions biochimiques se déroulant au niveau de la cellule. On peut ainsi voir les séquences biologiques comme le « langage de la vie ». C'est donc tout naturellement qu'on a tenté d'appliquer les outils de l'analyse linguistique à ces séquences.

A l'image de l'apprentissage d'une langue naturelle, il faut extraire de ces séquences des mots, des phrases, des structures, des grammaires afin d'être capable d'en comprendre le sens. Cette métaphore linguistique a notamment été décrite par Searls [Sea02].

Les signatures évoquées dans le chapitre précédent sont des représentations possibles de langages. De nombreux travaux utilisent la notion de grammaire afin de représenter tout ou partie des séquences ou structures macromoléculaires [Hea87, CJS06, SBU+94, AM97, SB02]. Nos propres travaux se rattachent à cette approche et nous nous sommes tournés vers des modèles de grammaires suffisamment expressifs pour représenter des interactions à distance.

Dans ce chapitre, nous commençons par définir les notions de langages et grammaires avant de présenter les difficultés et le cadre théorique de l'apprentissage de ceux-ci. Puis, nous présentons un bref état de l'art des techniques utilisées pour l'apprentissage de grammaires régulières modélisant des familles de protéines, ainsi que les limites de telles représentations.

2.1 Apprendre un langage

Le concept d'inférence grammaticale est apparu dès les années 50-60, c'est-à-dire à l'époque où Chomsky a formalisé la notion de langage [Cho57]. L'inférence grammaticale consiste à apprendre, à partir de données séquentielles ou structurées, une grammaire qui explique ces données, autrement dit qui permet de générer le langage auquel appartient ces données. Nous présentons ici les définitions et concepts liés à l'apprentissage d'un

langage à partir d'exemples.

2.1.1 Langages et grammaires, quelques définitions

On peut définir un langage comme un ensemble de mots sur un alphabet donné. Nous introduisons ici les définitions précises de ces concepts et les notations utilisées au travers des différentes représentations de la théorie des langages.

Définition 2.1 (Alphabet, mots et langages) *Un alphabet Σ est un ensemble fini non-vide de symboles. La taille de Σ , notée $|\Sigma|$ est le cardinal de l'ensemble.*
Un mot w sur un alphabet Σ est une suite $x_1...x_n$ finie de lettres x_i de Σ . Un mot est aussi nommé séquence ou chaîne. Le mot vide (défini par l'absence de symbole) est noté λ. L'ensemble des mots possibles sur Σ est noté Σ^*. Sa taille est notée $|w|$.
Un langage (formel) est un ensemble de mots.

Nous introduisons maintenant les définitions de facteurs (définition 2.2) et de contextes (définition 2.3) d'un langage, qui seront au cœur de nos travaux.

Définition 2.2 (Facteur) *Soit un alphabet Σ et un langage L, on nommera facteur (séquence de symboles), un élément $w \in \Sigma^*$ tel que $l,r \in \Sigma^*$ et $lwr \in L$.*

Définition 2.3 (Contexte) *Soit un alphabet Σ et un langage L, on nommera contexte une paire de facteurs $(l,r) \in \Sigma^* \times \Sigma^*$. Un contexte appartient au langage L s'il existe un facteur w tel que $lwr \in L$. On notera que $(l,r) \circ w$ le facteur lwr.*

La théorie des langages s'est particulièrement intéressée aux modèles permettant de représenter et définir un langage. Nous introduisons ici les représentations par grammaires.

Définition 2.4 (Grammaire formelle) *La grammaire d'un langage est constituée d'un quadruplet (Σ,N,R,S) où Σ est l'ensemble des symboles terminaux (alphabet), N l'ensemble des symboles non terminaux, R l'ensemble des règles de la forme $(N \cup \Sigma)^* \rightarrow (N \cup \Sigma)^*$ et S $(S \in N)$ l'axiome de départ. Le langage généré par une grammaire G est dénoté $\mathscr{L}(G)$.*

Chomsky[Cho57] définit quatre types principaux de grammaires hiérarchiquement imbriquées. On trouve, de la plus expressive à la moins expressive :

- les grammaires générales qui représentent des langages récursivement énumérables. Il n'existe pas d'algorithme général permettant de décider en temps fini à partir d'une telle grammaire si un mot quelconque appartient ou non au langage ;
- les grammaires contextuelles qui sont telles que le remplacement d'un élément non-terminal peut dépendre des éléments autour de lui : son contexte. Ses règles sont de la forme $sXt \rightarrow swt$ où $s,w,t \in (N \cup \Sigma)*$, $X \in N$ et $w \in (N \cup \Sigma)^+$. Le problème de l'appartenance d'un mot au langage à partir d'une grammaire contextuelle est NP-complet ;

- les grammaires hors-contexte (dites aussi algébriques) qui traitent les éléments non-terminaux individuellement (sans dépendance contextuelle). Ses règles sont de la forme $X \rightarrow w$ où $X \in N$ et $w \in (N \cup \Sigma)*$. L'appartenance d'un mot w à un langage à partir d'une grammaire algébrique peut-être calculée en un temps polynomial par rapport à $|w|$;
- les grammaires régulières où les règles sont de la forme : $X \rightarrow Aa$, $X \rightarrow a$ où $X, A \in N$ et $a \in \Sigma$ (grammaire linéaire gauche). Une grammaire régulière décrit la reconnaissance des mots caractère par caractère. La reconnaissance d'un mot w se fait en temps linéaire par rapport à $|w|$.

Nous nous interessons dans cette thèse plus particulièrement aux grammaires hors-contexte. Une séquence $\delta A \gamma$ de $(N \cup \Sigma)^+$ peut être dérivée en $\delta \alpha \gamma$, noté $\delta A \gamma \Rightarrow_G \delta \alpha \gamma$ s'il existe une règle de production $A \rightarrow \alpha$ dans R. La fermeture transitive de $\Rightarrow_G$ est notée $\overset{+}{\Rightarrow}_G$ et sa fermeture réflexive transitive $\overset{*}{\Rightarrow}_G$. L'ensemble des chaînes qui peuvent être dérivées de S est l'ensemble des formes sententielles $\{\alpha \in (N \cup \Sigma)^* : S \overset{*}{\Rightarrow}_G \alpha\}$ et le langage défini par une grammaire hors-contexte G est l'ensemble des formes sententielles composées uniquement de symboles terminaux $L(G) = \{w \in \Sigma^* : S \overset{*}{\Rightarrow}_G w\}$.

Un mot m appartient à $\mathscr{L}(G)$ s'il est *dérivable* par G, autrement dit s'il existe une suite de règles applicables à partir de l'axiome qui permettent de générer m. L'analyse d'un mot m par par une grammaire hors-contexte G est représentable sous la forme d'un *arbre de dérivation* dont les nœuds sont des non terminaux et les feuilles des symbole de Σ.

Exemple de grammaire hors-contexte

S → **Subject Verb Complement**

Subject → Mr Smith | He

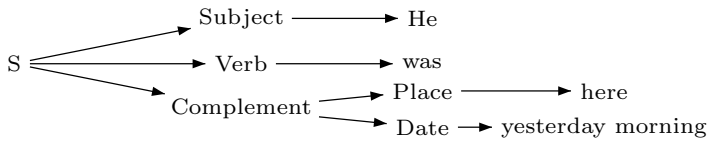
Verb → was | went | will be | will be gone

Complement → **Place Date** | **Date**

Place → there | here

Date → yesterday morning | tomorrow morning | tomorrow evening

La séquence "He was here yesterday morning" peut être dérivée par cette grammaire et l'arbre de dérivation est le suivant :



L'ensemble des mots pouvant être dérivés par un non terminal A est dénoté $yield(A)$. Ainsi, pour une grammaire $G = (\Sigma, N, R, S)$, $yield(S) = \mathscr{L}(G)$.

Il est à noter qu'il existe une formale normale permettant de représenter une grammaire hors-contexte : la forme normale de Chomsky (CNF). Une grammaire hors-contexte est

dite sous CNF ssi toutes ses règles de production sont de la forme $A \rightarrow BC$, $A \rightarrow a$, $A->\lambda$ où $A,B,C \in N$ et $a \in \Sigma$.

2.1.2 Cadre théorique de l'apprentissage d'un langage

Afin d'apprendre le langage de l'ensemble des séquences d'une même famille fonctionnelle, il faut extraire les caractéristiques au niveau séquentiel de cette fonction. On généralise ainsi l'ensemble d'apprentissage ce qui permet de reconnaître des séquences qui n'ont encore jamais été associées à la fonction cible. La généralisation est un mode de raisonnement très naturel à la base de l'apprentissage. Elle permet de déduire des structures, des règles de comportements à partir d'exemples (et parfois de contre exemples).

Dans le cas de nos familles de séquences protéiques, le principe de généralisation choisi devrait nous permettre, en théorie, de découvrir toutes les séquences appartenant au langage d'une famille.

Soit S_0 l'ensemble des séquences d'apprentissage (validées expérimentalement comme possédant une certaine fonction), L l'ensemble des séquences possédant la même fonction que S_0 et α un principe de généralisation, on peut définir l'ensemble des séquences atteignables par ce principe comme $S_n = \alpha(S_0)$. Dans un monde idéal, $S_n = L$. Mais comment savoir si S_n est bien le langage qu'on souhaitait obtenir au départ : le langage cible ? Comment juger de la qualité du langage appris quand on ne le connaît pas au départ ? Le langage cible est-il même apprenable à partir d'exemples positifs ?

Par exemple, la découverte d'un motif identique dans deux séquences d'enzymes possédant la même fonction ou la détermination d'une proportion commune de certains acides aminés sont deux principes de généralisation. Ils produiront alors un langage différent. Quel principe choisir ? La réponse à cette question dépendra de la complexité des données à modéliser. Pour nos séquences d'enzymes, nous avons cherché un principe permettant de prendre en compte des corrélations entre différentes parties d'une séquence.

2.1.3 La généralisation comme recherche dans un espace d'hypothèses

Le choix du principe d'induction est lié à la représentation du langage utilisée 2.1. Dans le cas des motifs Prosite [HBB+06], le langage généré par ces motifs est une sous classe des langages réguliers facilement représentable par des expressions régulières. Pour les langages, il existe essentiellement trois types de représentations : les grammaires, les machines et les expressions.

Dans le cadre de l'inférence grammaticale, les hypothèses sont généralement représentées par des grammaires (dans le cas des langages réguliers, automates et grammaires sont très proches) que l'on peut ordonner selon un ordre partiel de généralité et se représente sous forme d'un treillis comme sur la figure 2.1.

Dans notre cas, chaque nœud représente une hypothèse composé d'un ensemble de grammaires équivalentes, générant le même langage. Une hypothèse H_1 est dite plus *générale* que H_2 si le langage généré par H_1 est un sous-ensemble du langage généré par H_2.

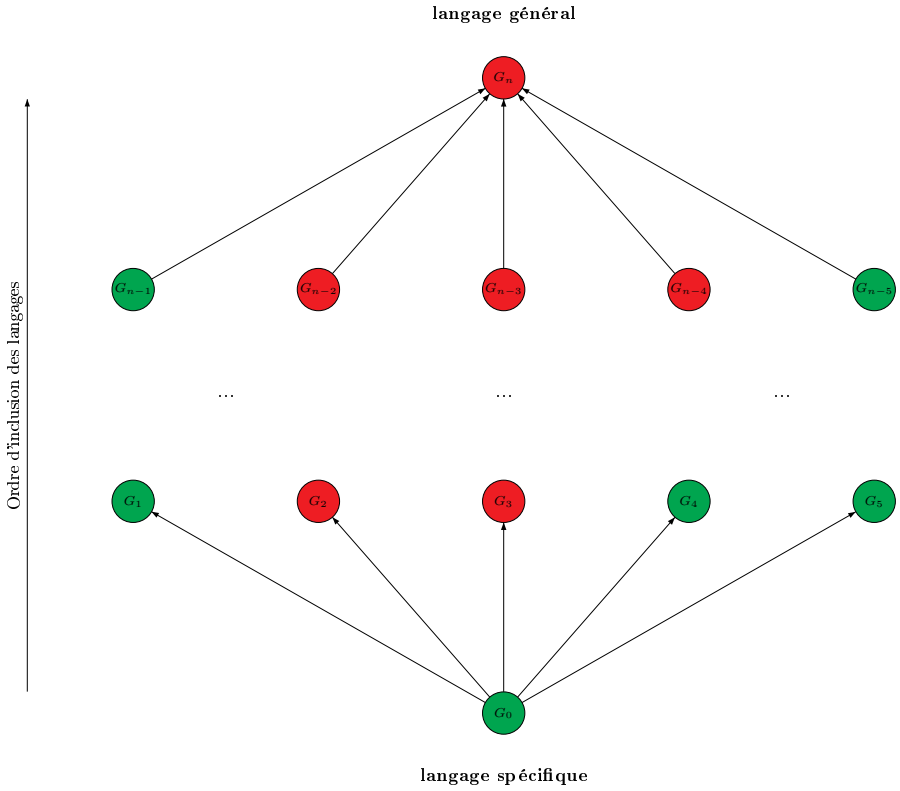


Fig. 2.1 – Représentation de l'espace d'hypothèses comme ordre partiel basé sur l'inclusion des langages générés par chaque concept

On verra qu'il est plus difficile de naviguer dans cet espace d'hypothèses dans le cas de grammaires algébriques étant donné l'indécidabilité de l'équivalence de deux grammaires.

On cherche alors l'ensemble des grammaires *consistantes* dans cet espace. Dans le cadre d'une inférence avec exemples positifs seulement, une hypothèse *consistante* est une hypothèse qui contient les données d'apprentissage.

Dans le cas où on ne possède pas d'exemples négatifs, rien n'empêche d'obtenir une grammaire qui reconnaîtrait le langage universel et le problème majeur est d'éviter la *surgénéralisation*. Pour cela, il existe plusieurs méthodes qui permettent de limiter cet effet : proposer une heuristique permettant de limiter la généralisation, structurer les

données *a priori* ou encore limiter l'apprentissage à une sous-classe de langages avec de bonnes propriétés.

On peut encore se demander comment choisir une solution optimale qui reconnaît le langage cible, c'est à dire comment choisir entre deux grammaires équivalentes appartenant au même concept ?

Pour sélectionner les solutions de manière efficace, il est nécessaire de posséder un ordre sur l'ensemble des hypothèses afin de déterminer la *meilleure* grammaire. Pour cela, on utilise souvent un *biais d'induction* [Mit80, GD95] qui permet de définir une fonction permettant de *mesurer* le caractère optimal d'une solution (au sens du biais introduit) afin d'obtenir un ordre total sur l'espace d'hypothèses et d'être capable de sélectionner la meilleure par rapport à ce principe.

Il existe plusieurs biais d'induction, la plupart sont basés sur le principe de simplicité du *Rasoir d'Ockham* qui pose que les hypothèses les plus simples sont les plus vraisemblables.

[CV03] présente un résumé des différents biais utilisés découlant de ce principe. La simplicité peut alors être vu comme une fonction mathématique qui vise à optimiser un critère donné.

Dans le cadre de l'inférence grammaticale, ce critère peut être basé sur le fait d'obtenir une grammaire de taille minimale ou encore sur un principe de déterminisation des règles évitant la production d'un mot par plusieurs arbres de dérivation.

Un principe fréquemment utilisé est le principe MDL (*Minimum Description Length*) [Ris78] basé sur le fait que la représentation qui permet la meilleure compression des données sera la solution optimale. Le principe MDL postule qu'il existe souvent une structure sous-jacente pouvant être utilisée pour compresser l'ensemble de données d'apprentissage. Il permet alors de découvrir cette structure [Grü94].

Une autre méthode est de choisir la solution qui minimise l'erreur empirique (ERM) [Vap92]. Pour cela il faut être capable de tester les solutions obtenues sur un autre ensemble appartenant ou non au langage cible, afin de déterminer le taux d'erreur de classification sur cet ensemble. L'hypothèse qui minimise l'erreur obtenue est alors considérée comme la solution optimale.

Ces approches se veulent plutôt empiriques.

Nous définissons ci-après les cadres d'apprentissage plus formels les plus utilisés permettant de définir si une classe de langages est apprenable à partir d'exemples, et s'il existe une grammaire minimale (ou canonique) permettant de représenter les langages de cette classe.

2.1.4 Apprenabilité : le cadre de l'identification à la limite

Il existe plusieurs cadres d'apprentissage qui ont permis de progresser dans la notion de convergence vers une cible d'apprentissage comme le cadre PAC [Val84] ou celui de l'apprentissage actif [Ang87]. Nous focaliserons ce travail sur les paradigmes d'identification à la limite. Ce cadre d'apprentissage fut développé en premier lieu dans [Gol67].

Le protocole d'identification à la limite est défini comme suit. Soit α une méthode d'inférence et $S = \{x_1, x_2, ...\}$ un ensemble d'exemples d'un langage L. La méthode α

considère les exemples dans l'ordre des indices et on note S_i le sous-ensemble fini des i premiers éléments de S. Elle propose à chaque étape une hypothèse $\mathscr{L}(\alpha(S_i))$. La méthode α identifie à la limite le langage L s'il existe un indice fini j tel que $\mathscr{L}(\alpha(S_i)) = \mathscr{L}(\alpha(S_j))$ pour tout $i \geq j$ et que $\mathscr{L}(\alpha(S_j)) = L$.

Ce paradigme, incrémental, est très éloigné de la pratique où on dispose d'un échantillon de données, c'est-à-dire un ensemble fini de données d'apprentissage, et non une séquence infinie. C'est pourquoi le cadre de l'identification à la limite par données fixées a été introduit [Gol78]. Il repose sur la définition d'un échantillon caractéristique.

S est un *échantillon caractéristique* d'un langage L, s'il illustre une méthode d'inférence α telle que, pour tout ensemble de données S_0 tel que $S \subseteq S_0$, $\mathscr{L}(\alpha(S_0)) = L$.

Autrement dit, un échantillon caractéristique est un échantillon suffisant pour assurer la convergence de l'apprentissage vers la solution. Il est essentiel d'avoir les bonnes données d'apprentissage pour arriver au modèle espéré.

Cette définition d'échantillon caractéristique permet alors de définir l'identification à la limite par données fixées. Une classe de langages est *identifiable à la limite par données fixées* si pour tous ses langages L, pour une méthode d'inférence α, il existe un échantillon caractéristique $S \subseteq L$ tel que $\mathscr{L}(\alpha(S)) = L$.

Le problème de cette définition, c'est qu'elle n'exige rien sur la taille des données d'entrée, de la représentation ou sur le temps de convergence de l'algorithme qui mène à la solution. Pour une convergence pratique, il est nécessaire de préciser d'avantage le cadre.

La première définition d'identification polynomiale à la limite a été donnée dans [Pit89]. La plus récente [dlH97] dit qu'une classe de langages $\mathscr{L}$ est *identifiable en temps et données polynomiaux* si pour tout langage $L \in \mathscr{L}$ et à partir d'une classe de représentation $\mathscr{R}$, à partir d'un ensemble d'apprentissage S_0 de taille m, il existe un algorithme α et deux polynômes $p()$ et $q()$ tel que α retourne une solution correcte H dans $\mathscr{R}$ en un temps $O(p(m))$ et que pour toute représentation $R \in \mathscr{R}$ de taille k d'un langage $L \in \mathscr{L}$, il existe un échantillon caractéristique de taille $O(q(k))$.

2.1.5 Validation du langage appris

Les définitions précédentes permettent de démontrer formellement l'apprenabilité théorique du langage et de sa représentation que nous chercherons à apprendre via un principe de généralisation particulier.

En pratique, il n'est pas certain que la classe de langages considérée et/ou les choix d'induction dans l'espace de recherche soit suffisante pour permettre l'identification du langage cible sous-jacent au exemples positifs. Le langage appris de cette façon permettra-t-il réellement de discriminer les différentes classes enzymatiques ? Pour le savoir, il faut recourir à une évaluation sur des données réelles afin de vérifier que le langage identifié a un certain pouvoir prédictif par rapport aux séquences observées.

On peut ainsi définir plusieurs étapes à l'apprentissage d'un langage puis à sa validation. Les différentes étapes du processus à partir d'un ensemble de séquences consisteront à : définir une représentation et un principe d'induction, vérifier formellement que le langage

appris est consistant et converge vers un langage solution, et enfin, vérifier empiriquement que le langage appris est adéquat pour représenter nos données.

Dans la section suivante, nous nous intéressons aux travaux d'apprentissage de grammaires régulières pour modéliser des séquences de protéines.

2.2 Inférence de grammaires régulières

L'inférence grammaticale a pour but d'apprendre une grammaire qui constitue une représentation possible d'un langage, à partir d'exemples, et parfois de contre-exemples, de ce langage. Dans le cas qui nous concerne, les séquences de protéines d'une même famille déjà annotées sont les exemples appartenant à un langage inconnu et la généralisation doit produire le langage complet de cette famille, représenté par la grammaire.

2.2.1 État de l'art

Un des problèmes les plus étudiés en inférence grammaticale est l'apprentissage d'automates déterministes à états finis à partir de données positives et négatives.

Les automates (définition 2.5 et 2.6) sont des représentations graphiques équivalentes à la classe des langages réguliers qui est la plus haute dans la classification de Chomsky qui soit polynomialement identifiable à partir de données fixées [dlH97].

Définition 2.5 (NFA) *Un automate fini non déterministe(NFA), est un quintuplet* $A = (\Sigma, Q, I, \delta, F)$ *tel que :*

- Σ *est l'alphabet d'entrée*
- Q *est l'ensemble fini d'états*
- $I \in Q$ *est l'ensemble des états initiaux*
- δ *est la fonction de transition de l'automate définie de* $Q \times \Sigma$ *vers* 2^Q*, un triplet* (q, a, q') *avec* $\delta(q, a) \in Q'$ *est appelé transition*
- F *est l'ensemble des états finals*

Définition 2.6 (DFA) *Un automate fini déterministe(DFA), est un quintuplet* $A = (\Sigma, Q, I, \delta, F)$ *tel que :*

- Σ *est l'alphabet d'entrée*
- Q *est l'ensemble fini d'états*
- $I \in Q$ *est l'ensemble des états initiaux*
- δ *est la fonction de transition de l'automate définie de* $Q \times \Sigma$ *vers* Q*, un triplet* (q, a, q') *avec* $Q' \in \delta(q, a)$ *est appelé transition*
- F *est l'ensemble des états finals*

Une famille d'algorithmes classiques a pour opérateur de généralisation les *fusions d'états* à partir d'un automate canonique maximal, noté MCA. Le MCA est simplement l'union des automates canoniques représentant chaque mot de l'échantillon d'apprentissage. Le MCA des séquences positives représente exactement le langage des exemples

donnés. L'idée de l'apprentissage par fusion d'états est de généraliser le langage (augmenter le nombre de séquences acceptés). En effet, les fusions d'état génèrent de nouveaux automates introduisant de nouveaux chemins et donc de nouvelles séquences dans le langage, à partir du MCA. Les différents algorithmes diffèrent ensuite par leur stratégie de choix d'états à fusionner et leur critère d'arrêt. Les algorithmes utilisant la fusion d'état tel que l'algorithme RPNI, dans [OG92] ou [Lan92], sont capables de généraliser et d'identifier polynomialement la classe des langages réguliers à la limite. Un espace d'hypothèses en treillis a été présenté dans [DMV94] : les nœuds du treillis correspondent aux différents automates qui peuvent être obtenus en fusionnant les états de l'automate canonique maximal. Le nombre de nœuds, et par conséquent la taille de l'espace d'hypothèses, est exponentielle par rapport à la taille initiale du MCA. Cette représentation de l'espace d'hypothèses fournit des moyens de prouver la convergence de certains algorithmes.

Le problème de l'apprentissage à partir de données positives seulement est probablement le problème d'inférence grammaticale le plus intéressant en pratique. Comme nous l'avons vu, il existe deux types de méthodes : celles utilisant des heuristiques et celles permettant une identification à la limite. Cette deuxième méthode est néanmoins freinée par le résultat de Gold [Gol67] qui dit qu'aucune classe de langages au delà des langages finis n'est identifiable à la limite à partir de données positives seulement à cause du problème de surgénéralisation.

Plusieurs approches ont alors été adoptées pour controler la généralisation : limiter la représentation des langages à une sous-classe, utiliser un critère statistique pour stopper les fusions ou utiliser des contre-exemples.

Angluin [Ang82] définit la classes des langages réversibles et propose un algorithme pour cette sous-classe des langages réguliers. Angluin donne d'autres résultats théoriques généraux permettant d'identifier des sous-classes de langages réguliers à partir d'exemples positifs seulement [Ang80].

[GVO90] propose lui un algorithme pour la classe des langages k-testables. Dans la même ligne de recherche, [KMT97] propose un algorithme permettant l'identification des langages linéaires équilibrés et [DLT02] présente des sous-classes de langages réguliers pouvant être identifiés par des automates finis non-déterministes à partir d'exemples positifs seulement.

2.2.2 Application aux séquences de protéines

On peut appliquer les principes de la théorie des langages aux séquences d'acides aminés en faisant les choix suivants :

- L'alphabet est composé de 20 lettres (acides aminés) : A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V.
- Les mots sont les séquences composées à partir de cet alphabet.

Lorsqu'il s'agit de séquences de protéines, il est important de tenir compte des similitudes entre les 20 acides aminés (l'alphabet σ_a des protéines) découlant des propriétés physico-chimiques partagées par ceux-ci : certains remplacements d'acides aminés n'ont pas d'impact tandis que d'autres ont un effet sur la fonction ou la structure de la protéine.

Pour tenir compte de cette connaissance directement dans les séquences d'apprentissage, une approche standard consiste à recoder les protéines sur un alphabet plus petit fondé sur leurs propriétés, tels que l'indice d'hydropathie ou le codage de Dayhoff ([YIK94], [PLCS06] et [PLC08]).

On trouve plusieurs références concernant l'application de l'inférence grammaticale à des séquences de protéines qui permettent de découvrir certaines structures spécifiques en utilisant des connaissances $a\ priori$ sur les structures 3D [YIK94, DN09, PLCS06, PLC08, DN09].

Ainsi, une famille de séquences d'enzymes peut-être vue comme un langage. On peut définir un langage L qui contient l'ensemble des séquences possédant une fonction commune f et déterminer si une nouvelle séquence possède la fonction f selon qu'elle appartienne ou non au langage L. Le problème est d'identifier ce langage à partir d'exemples positifs, c'est-à-dire d'un échantillon de séquences connues possédant la fonction f.

L'apprentissage d'automate à états finis dans ce but a fait l'objet de plusieurs recherches [Fre03, Ler05]

Ces travaux ont notamment montré que l'emploi d'une représentation non-déterministe est plus adapté à cette tâche. [Ler05] a montré que les principales heuristiques concernant le choix et l'ordre des états à fusionner dans la plupart des algorithmes mentionnés ci-dessus n'étaient pas adaptés à la généralisation de séquences de protéines. Cela est dû au fait que ces heuristiques fusionnent en priorité les états initiaux, ou terminaux, du MCA. Cette écriture introduit une heuristique de fusion intéressante dirigée par la similarité des sous-séquences de l'échantillon d'apprentissage mais semble obtenir des modèles trop spécifiques.

Dans la même lignée, [Ker08] présente une méthode d'inférence grammaticale basée sur les algorithmes à fusions d'états et sur une heuristique biologique de conservation de séquences. Une famille est alors représentée par un automate fini non déterministe. Nous terminons cette section en présentant cette méthode dont nous sommes partis pour réaliser nos travaux.

2.2.3 Protomata-Learner

L'algorithme Protomata-Learner commence par une phase d'alignement multiple local partiel que nous avons décrit au chapitre 1. La fusion des états lors de la création de l'automate est alors guidée par les alignements découverts pour obtenir une généralisation de l'automate canonique maximal prenant en compte les zones caractéristiques des séquences.

Alignement multiple local et partiel Cette méthode commence par aligner l'ensemble des séquences d'apprentissage dans un alignement multiple local et partiel.

Un ensemble de sous-séquences similaires est appelé un bloc PLMA. Une fois l'alignement produit, on peut produire un automate maximal canonique correspondant (exemple figure 2.2).

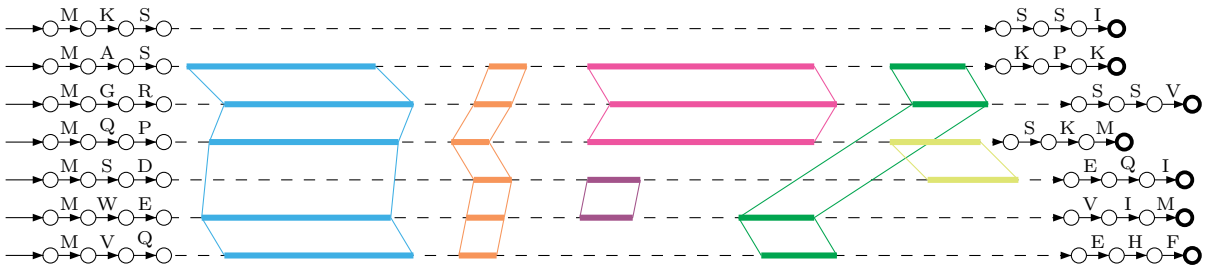


Fig. 2.2 – Représentation du MCA généré à partir des séquences sur lesquelles nous avons projeté un alignement multiple local partiel

Généralisation par fusion d'états Ensuite, afin de généraliser le MCA, on procède à la fusion des séquences d'états correspondant aux blocs PLMA, ce qui crée de nouveaux chemins dans l'automate qui accepte ainsi de nouvelles séquences.

Les états entre les zones reconnues comme caractéristiques sont fusionnés en un seul état acceptant une séquence quelconque. En effet, ces zones étant peu informatives, il est possible d'y trouver des sous-séquences très différentes les unes des autres. Ces états sont appelés des états de gap.

L'algorithme détecte aussi les zones dites d'exception. En effet, certaines séquences exceptions forment des chemins permettant de court-circuiter les blocs et la fusion de ceux-ci comme états de gap entraînerait une généralisation à l'excès. Les états correspondant à ces zones ne sont donc pas fusionnés.

La généralisation de l'automate de la figure 2.2 est présentée en figure 2.3.

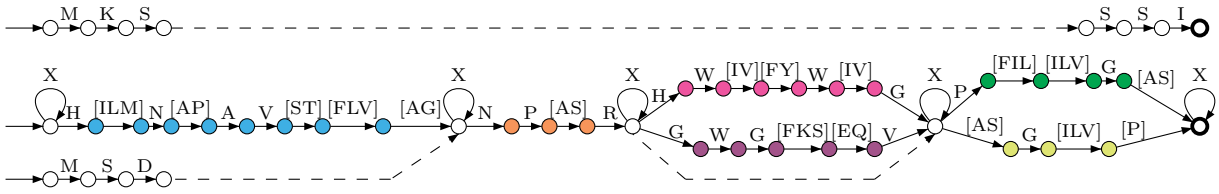


Fig. 2.3 – Automate après fusions d'états

Généralisation des transitions Enfin, il est également possible de généraliser certaines transitions selon les propriétés physico-chimiques des acides aminées présents sur cette transition. En effet, si les transitions possibles d'un état à l'autre sont multiples et font parties d'un même groupe physico-chimique (cf figure 2.4), on peut généraliser ces transitions au groupe d'acides aminés auquel elles appartiennent . Au contraire, si les transitions

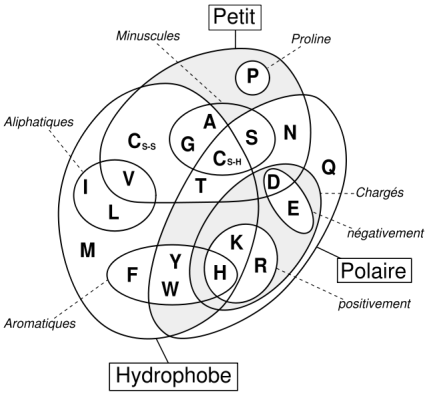


Fig. 2.4 – Propriétés physico-chimiques des acides aminés

sont nombreuses et n'appartiennent pas à un groupe précis, on généralise pour autoriser la transition par n'importe quelle lettre de l'alphabet.

L'automate obtenu définit alors les séquences d'une famille de protéines comme un enchaînement de blocs de conservation. Il autorise des chemins de gaps, permettant de décrire des zones peu similaires acceptant des séquences quelconques, ainsi que des chemins d'exceptions.

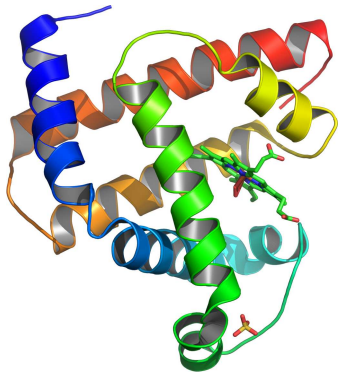
2.2.4 Discussion

L'inférence de langages réguliers permet de représenter la structure linéaire de la protéine comme enchainement de motifs séparés par des gaps. Cependant, il a été établi que les repliements spatiaux des protéines seraient apparemment mieux conservés au sein d'une même famille que la séquence elle même [CL86a]. De plus, c'est la structure qui porte la fonction de la protéine. Ainsi, il serait intéressant de pouvoir aussi apprendre des interactions entre différentes zones d'une séquence en découvrant des motifs liés plus ou moins éloignés. Sur la figure 2.5, on peut ainsi voir que des interactions peuvent exister entre des parties de séquences a priori éloignées dans la structure primaire mais proches en trois dimensions.

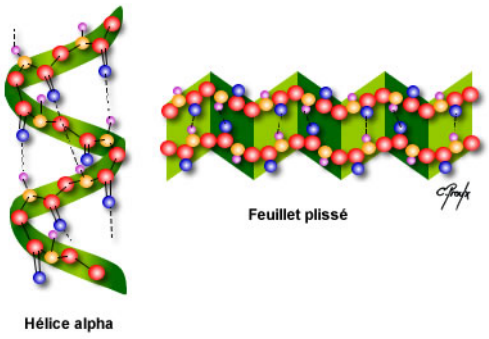
Deux types de corrélations doivent être repérées :

- à courte distance (figure 2.5(b)), à l'intérieur d'un même bloc, comme dans les hélices alpha.
- à longue distance (figure 2.5(a)), dans des blocs différents, comme par exemple l'hélice rouge et l'hélice verte qui sont éloignées dans la structure primaire mais proche en trois dimensions.

Les langages réguliers ne peuvent pas représenter ces types de corrélations à cause de leur linéarité. Pour cette raison, nous nous sommes intéressés à l'extension des techniques d'inférence à la classe des langages algébriques.



(a) Une séquence protéique en 3D

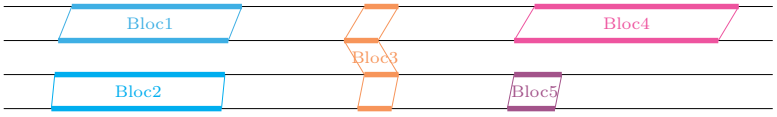


(b) Quelques repliements fréquents (hélice alpha et feuillet béta)

Fig. 2.5 – Exemples de structures d'une protéine

Nous illustrons l'intérêt,en terme de gain d'expressivité, de passer du régulier à l'algébrique sur un exemple (figure 2.6).

On remarque dans cet exemple que la modélisation des séquences par une grammaire linéaire droite produit un effet $reset$. Il est impossible de décrire des corrélations entre les blocs 1,2,3 et 4 puisqu'une fois l'état commun dépassé la grammaire ne garde pas en mémoire le chemin suivi. En réalité, il serait possible de le représenter avec une grammaire régulière mais on perdrait alors en compacité de représentation et la généralisation reviendrait au mieux à la première grammaire ou sera inexistante. Avec une grammaire algébrique, nous serons par exemple facilement capables d'exprimer que le bloc 1 et le bloc 4 vont de paire. On voit sur sur la figure 2.6 qu'une grammaire de ce type est capable d'exprimer les interactions entre les blocs de façon compacte, sans perdre en expressivité. La question est maintenant de savoir si le langage généré par cette représentation peut être généralisé. C'est ce que nous allons essayer de déterminer dans les chapitres suivants.



(a) Séquences ayant un bloc de conservation commun

S ⟶ Bloc1 **S1** | Bloc2 **S1** **S1** ⟶ Bloc3 **S2** **S2** ⟶ Bloc4 | Bloc5	**S** ⟶ Bloc1 **S1** | Bloc2 **S2** **S1** ⟶ Bloc3 **S3** **S2** ⟶ Bloc3 **S4** **S3** ⟶ Bloc4 **S4** ⟶ Bloc5
(b) Modélisation avec une grammaire régulière compacte (effet reset)	(c) Modélisation avec une grammaire régulière plus expressive

S ⟶ Bloc1 **S1** Bloc4 | Bloc2 **S1** Bloc5
S1 ⟶ Bloc3

(d) Modélisation avec une grammaire algébrique compacte et expressive

Fig. 2.6 – Compacité vs Expressivité

Chapitre 3

Apprentissage de grammaires par substituabilité

Les grammaires algébriques ont un pouvoir d'expressivité supérieur à celui des langages réguliers. L'espace de recherche correspondant en apprentissage est également beaucoup plus complexe. Il faut alors restreindre la recherche à des sous-classes de langages algébriques ou utiliser des heuristiques tout en conservant un bon pouvoir prédictif sur des applications pratiques par l'utilisation d'un principe de généralisation naturel.

Dans ce chapitre, après avoir présenté les principaux algorithmes traitant ce problème, nous nous intéressons au principe de généralisation qu'est la substituabilité et qui semble être sous-jacent à toutes les méthodes efficaces. Nous étudions plus particulièrement le problème de l'inférence de la sous-classe des langages substituables développée par [CE07] et adaptons ce procédé pour le cas des séquences protéiques. Nous introduisons de nouvelles classes de langages substituables, de nouveaux critères de généralisation associés ainsi qu'un algorithme capable de générer une grammaire réduite conservant la structuration des exemples de l'échantillon d'apprentissage. Nous terminons en présentant les expériences que nous avons effectuées sur des ensembles de séquences protéiques.

3.1 Apprentissage de grammaires algébriques

Dans cette section, nous présentons les difficultés liées à l'inférence grammaticale de langages algébriques, lié au fait qu'il faut non seulement apprendre la grammaire générant le langage voulu mais aussi la structuration de celui-ci.

Nous avons alors cherché dans l'état de l'art de l'inférence de grammaires algébriques des algorithmes non supervisés, fonctionnant à partir d'exemples positifs seulement et à visée applicative. Le traitement automatique des langues est un domaine où il y a eu des avancées significatives en apprentissage ces dernières années. Nous présentons les principaux travaux de ce domaine dont le principe de généralisation semble lié à la découverte de la structuration des exemples par substituabilité. Ces approches semblent être les plus intéressantes pour notre problème bien qu'elles doivent être adaptées pour le traitement

de séquences protéiques. Elles ont montré en particulier des résultats encourageants sur des corpus réels.

3.1.1 Les difficultés de l'apprentissage de grammaires algébriques

Dans un premier temps, nous expliquons quelles sont les difficultés principales de l'apprentissage d'une grammaire algébrique en nous inspirant de la thèse de R. Eyraud [Eyr06]. Nous mettrons l'accent sur quatre des difficultés évoquées.

Opérabilité

Un des premiers problèmes rencontrés dans l'inférence de grammaires algébriques, comparée à l'inférence régulière, découle de la faiblesse des propriétés liées à la classe de langages apprise. Ainsi, l'union, la concaténation et l'intersection de langages réguliers, ou le complémentaire d'un langage régulier sont des langages réguliers. Ces propriétés fortes permettent d'utiliser certaines opérations lors de l'inférence sans sortir de la classe de langages visée.

Cependant, pour la classe des langages hors-contexte, la stabilité de ces opérations n'est pas toujours assurée. En effet, si l'union et la concaténation de deux langages hors-contexte forment un langage hors-contexte, ce n'est pas le cas de leur intersection ou du complémentaire.

Cet état de fait rend difficile l'introduction d'exemples négatifs, qui ne délimitent pas forcément un langage hors-contexte et conduit à inférer le langage à partir d'exemples positifs seulement, alors que le résultat de Gold a montré l'impossibilité de l'identification à la limite de langages hors-contexte à partir d'exemples positifs seulement [Gol78].

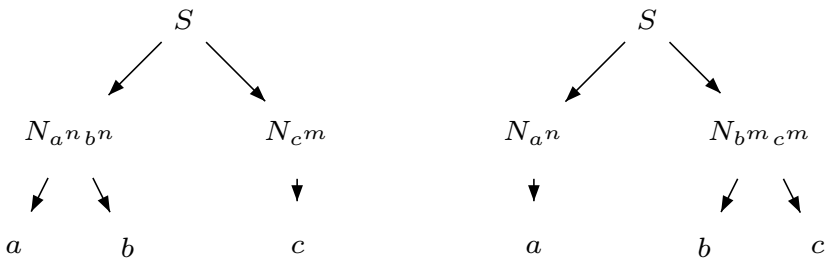
Indécidabilité

Un second problème provient de l'indécidabilité de l'équivalence de deux grammaires algébriques. En effet, il est dans la plupart des cas impossible de savoir si deux grammaires hors-contexte génèrent le même langage ce qui rend difficile la recherche d'une grammaire dans un espace d'hypothèses ordonné partiellement par rapport à l'inclusion des langages générés par ces grammaires.

Dans le cas de l'inférence de langages réguliers à l'aide d'automates, il existe un représentant canonique calculable pour chaque langage. On peut donc réduire cet apprentissage à l'identification de ce représentant. Dans le cas des grammaires hors-contexte, lorsqu'il existe un nombre infini de représentations, il est difficile de les élaguer puisque l'équivalence ne peut être prouvée dans le cas général.

L'identification doit donc être basée sur le langage lui-même et non sur sa représentation, ce qui rend dans le même temps impossible l'identification à la limite en temps et données polynomiaux de langages algébriques à partir d'exemples [dlH97].

Cependant, on peut noter que certaines sous-classes peuvent néanmoins être apprises s'il existe une grammaire canonique pour tous les langages de ces sous-classes et un



(a) Une dérivation possible de abc (b) Une autre dérivation possible de abc

Fig. 3.1 – Dérivations ambiguës du mot abc générées par une grammaire représentant le langage algébrique $\{a^n b^n c^m \mid n,m \geq 0\} \cup \{a^n b^m c^m \mid n,m \geq 0\}$. 3.1(a) privilégie la structure du sous langage $a^n b^n c^m$ alors que 3.1(b) privilégie le sous langage $a^n b^m c^m$.

algorithme capable d'apprendre cette dernière. De récents résultats de [Cla14] introduisent ainsi la notion de *strong learning* pour certaines sous-classes des langages algébriques.

Ambiguïté

Un autre problème majeur dans l'inférence de grammaires algébriques est l'ambiguïté inhérente à certains de ces langages. Des langages possèdent alors à plusieurs arbres de dérivation possibles.

Un exemple de langage inhéremment ambigu est par exemple le langage $\{a^n b^n c^m \mid n,m \geq 0\} \cup \{a^n b^m c^m \mid n,m \geq 0\}$. Les mots de la forme $a^n b^n c^n$ ont alors deux dérivations différentes [Ogd68]. Un exemple de dérivations du mot abc est montré en figure 3.1.

La déterminisation supprimerait une partie des arbres de dérivation possibles et certaines structures du langage, et au final certains mots du langage cible. Quand on sait que la déterminisation est un principe largement utilisé dans l'inférence d'automates, on conçoit que l'abandon de celui-ci dans le cas des grammaires algébriques ne permet pas la réalisation des algorithmes efficaces développés dans le cadre régulier.

Structuralité

Le dernier problème sur lequel nous mettrons l'accent est celui de la structuralité du langage appris par une grammaire algébrique. En effet, l'intérêt d'apprendre une représentation du langage, plutôt qu'un classifieur répondant à la question "ce mot appartient-il au langage ?" est que cette représentation apporte une information sur le langage lui-même.

Ainsi, alors que la représentation sous forme d'expressions ou d'automates pour les réguliers conserve l'information sur la structuration des exemples, les formes normales

souvent employées pour représenter les grammaires algébriques font perdre cette notion de structure induite par le langage cible.

Lorsque qu'on apprend un langage algébrique, il faut donc apprendre non seulement les mots acceptés par ce langage mais aussi les structures d'analyse associées, ce qui est une double difficulté.

Ce dernier point a fondé la plupart des approches d'apprentissage des langages algébriques.

3.1.2 Apprentissage de la structure d'un langage

Dans le cadre de l'apprentissage de langues naturelles, Chomsky [Cho64] propose une organisation du langage en deux niveaux. La *structure de surface*, qui correspond à la séquence elle-même, à l'énoncé produit. Selon la théorie générative, ce niveau qui détermine l'interprétation sémantique, est le résultat d'opérations complexes à partir de la *structure profonde*, un ensemble de règles qui définit l'agencement des constituants du langage. Un constituant est alors défini comme un élément fonctionnel dans une hiérarchie grammaticale.

Dans cette théorie, comprendre la structure profonde d'un langage à partir d'exemples permet d'en comprendre le sens. Ainsi, la grammaire générative se veut explicative dans le sens où elle doit chercher à comprendre l'organisation d'un langage afin de pouvoir formuler un ensemble infini de phrases. L'apprentissage porte alors non sur la production de séquences sémantiquement correctes en tant que telles, mais sur les mécanismes permettant la construction de ces séquences. La grammaire tente d'expliquer les règles que chacun applique de façon intuitive pour construire un discours cohérent.

A partir des années 1930, Bloomfield et Harris [Blo33, Har54] proposent d'apprendre la structure d'un langage grâce à l'analyse distributionnelle. Le distributionalisme est fondé sur une certaine hiérarchie entre les séquences de facteurs qui définissent une structure distributionnelle et sur la possibilité de substituer certains facteurs entre eux. On appelle ces facteurs des constituants du langage. Tous les constituants qui occupent le même environnement linguistique (le même contexte) sont en situation d'équivalence distributionnelle et appartiennent à la même catégorie grammaticale.

D'après [CRA76, Wol80], on distingue alors deux groupe de constituants : les constituants dit *paradigmatiques* et ceux dit *syntagmatiques*. Les premiers sont définis comme des groupes de mots dont la nature syntaxique est équivalente : ils peuvent être substitués dans un corpus. Par exemple, les mots "bleu" et "blanc" peuvent être substitués dans n'importe quelle phrase en français sans en altérer la correction syntaxique. Les deuxièmes constituent des groupes de mots dépendants au niveau de la syntaxe, ils forment des sous-structures à l'intérieur du langage, on peut prendre l'exemple d'un groupe nominal composé alors de plusieurs mots dépendants : un déterminant, un nom et un adjectif, c'est la détection de ces classes qui va déterminer la hiérarchie de la grammaire, l'arbre de dérivation des exemples. Deux constituants syntagmatiques peuvent former un groupe paradigmatique et se substituer s'il possèdent une nature syntaxique équivalente, comme deux groupes nominaux.

A partir de ces notions, un grand nombre de méthodes d'analyse de corpus a été développé pour mettre au jour les classes de mots et les patrons syntaxiques caractéristiques d'une langue. Dans les travaux de Harris, l'analyse et la normalisation syntaxiques sont effectuées à la main. Mais, dans la communauté du traitement automatique des langues, un certain nombre de travaux ont été menés pour exploiter le principe de l'analyse distributionnelle.

Ce principe a notamment été utilisé dans le cadre de l'analyse syntaxique non supervisée afin de trouver les catégories syntaxiques de mots dans un corpus et ainsi obtenir des annotations automatiques de textes, par exemple pour induire automatiquement des catégories grammaticales (*part of speech*) [Sch95, Cla03].

Dans le cadre de l'inférence de grammaires hors-contexte à partir d'exemples positifs seulement, on retrouve dans l'état de l'art des principes inspirés de la théorie distributionnelle. L'inférence est souvent réalisée en deux étapes : l'identification des constituants du langage défini par les exemples et l'inférence des règles qui définissent la hiérarchie des constituants.

Identification des constituants La construction de grammaires algébriques requiert deux types de décision : "quels facteurs sont considérés comme constituants ?" (principe syntagmatique) et "quels facteurs peuvent être substitués les uns aux autres ?" (principe paradigmatique).

La première question est presque toujours traitée grâce à un alignement des séquences d'exemples qui permet d'établir la fréquence d'utilisation d'un facteur. Ainsi, l'algorithme ABL [vZ00] avec les exemples suivants en entrée :

$$\begin{array}{c} she\ is\ smart \\ she\ is\ tired \end{array}$$

produit un alignement où les numéros associés à chaque parenthèse indiquent les limites de chaque constituant :

$$\begin{array}{c} (0\ she\ is\ (1\ smart)1\ )0 \\ (0\ she\ is\ (1\ tired)1\ )0 \end{array}$$

Chaque parenthésage fait alors l'objet d'un non-terminal (après avoir sélectionné les alignements non ambigus par rapport aux exemples données). L'algorithme Adios [SHRE05] après avoir aligné ses exemples à l'aide d'un graphe, utilise une mesure qui a pour effet de détecter les facteurs les plus fréquents et crée pour chacun un nouveau non terminal le produisant. E-grids [PPK+04] quant à lui dispose d'un opérateur $CreateNT$ qui crée un nouveau non-terminal si son utilisation permet une grammaire plus simple (dans le sens du principe MDL).

Le but de ces non-terminaux n'est souvent pas de généraliser les exemples mais de les compresser en trouvant leurs constituants, et donc une structure sous-jacente.

Dans un deuxième temps, on trouve des mécanismes cherchant déterminer les facteurs substituables et à créer un non-terminal permettant de générer des constituants appartenant à la même classe. Dans ABL, cette phase se fait dans le même temps que

la première car l'alignement permet de détecter les classes de mots possédant un même contexte. Dans [SHRE05], [PPK+04] ou encore [Cla01], le principe est toujours le même, détecter les facteurs apparaissant dans un contexte identique et créer un nouveau non-terminal dans la grammaire reconnaissant ces facteurs comme dans l'exemple figure 3.2.

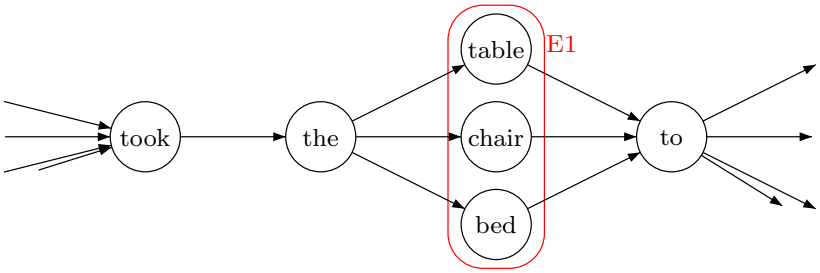


Fig. 3.2 – Détection d'une classe de mots substituables dans Adios suite à l'alignement d'exemples et la détection de facteurs apparaissant dans un contexte commun

La plupart de ces algorithmes présentent alors une phase de sélection des non-terminaux obtenus, basée par exemple sur le calcul de l'information mutuelle des éléments du contexte d'une classe [Cla01].

Ces non terminaux permettent alors de généraliser les exemples par substitution de facteurs, à n'importe quel endroit du texte, pourvu qu'ils aient été jugés comme syntaxiquement congruents car apparaissant dans un même contexte.

Inférence des règles Une fois les non-terminaux déterminés, arrive une phase de création de règles grammaticales.

Dans la plupart des cas, les différents algorithmes utilisent une approche *data-driven* et partent d'une grammaire maximale composée d'un axiome S générant exactement l'ensemble des exemples donnés en entrée (que l'on peut comparer à un MCA pour les automates présentés au chapitre précédent) où l'on remplace systématiquement chaque constituant par le non-terminal correspondant calculé à l'étape précédente. Une fois un facteur remplacé, il n'est plus possible de remplacer les facteurs découlant d'une ambiguïté et pour lesquels un non-terminal couvre déjà une partie du facteur.

Dans l'exemple de la figure 3.2, Adios répercute le nouveau non-terminal ainsi : la grammaire initiale

$$S \longrightarrow took\ the\ table\ to$$
$$S \longrightarrow took\ the\ chair\ to$$
$$S \longrightarrow took\ the\ bed\ to$$

se transforme en

$$S \rightarrow took\ the\ S1\ to$$
$$S1 \rightarrow table/chair/bed$$

Dans d'autres algorithmes, on préfère utiliser une approche permettant d'éviter la phase de sélection en conservant ainsi toutes les ambiguïtés du langage. Le but est alors de créer de nouvelles règles avec l'ensemble des non-terminaux obtenus précédemment. Pour cela, [CE07] choisit de se limiter à une représentation en forme normale de Chomsky (CNF) où les règles sont de la forme $A \rightarrow BC$ où $A, B, C \in N$. Ainsi, toutes les combinaisons possibles de triplets sont testées et l'existence de ces règles est vérifiée sur les données initiales. Nous en donnons un exemple plus loin (sous-section 3.1.3).

Évaluation des grammaires La majeure partie de ces algorithmes a été évaluée de façon expérimentale et comparée sur des corpus annotés, où les structures à découvrir sont clairement identifiées, afin de vérifier la validité des structures découvertes de façon non supervisée. Ils produisent des résultats encourageants qui montrent la pertinence de l'approche par analyse des éléments substituables.

Cependant, dû à la sélection des non terminaux à conserver et à la désambiguïsation des exemples opérée par la plupart des méthodes, il est impossible de prouver leur convergence vers une classe de langages donnée.

Nos propres travaux reposent sur ceux présentés dans [CE07], qui présentent un intérêt particulier du fait de la formalisation du principe de substituabilité et du résultat d'apprenabilité des langages hors-contexte substituables qui en découle.

Nous présentons donc dans ce qui suit ces principes et résultats plus en détails.

3.1.3 Concepts formels et formalisation du principe de substituabilité [Cla10b, Cla10a]

Analyse de concepts formels : quelques définitions Un *contexte formel* est un triplet $(\mathscr{O}, \mathscr{A}, I)$, où $\mathscr{O}$ est un ensemble fini d'objets, $\mathscr{A}$ un ensemble fini d'attributs et I est une relation entre les objets et les attributs. Il peut être représenté sous forme de table, un exemple est donné ci-après :

	A_1	A_2	A_3
O_1	X	X	X
O_2	X	X	X
O_3			X

Sur cet exemple, les objets sont dénotés O_i et les attributs A_i, la case $\langle O_i \times A_i \rangle$ contient "x" s'il existe une relation I entre O_i et A_i.

L'analyse de concepts formels [Wil82] définit intuitivement un concept comme correspondant à un rectangle maximal de la table formée par la relation binaire du contexte. On

définit les opérations suivantes : $O' = \{a \in \mathscr{A} \mid (o,a) \in I, \forall o \in \mathscr{O}\}$,

$$A' = \{o \in \mathscr{O} \mid (o,a) \in I, \forall a \in \mathscr{A}\}.$$

Intuitivement, O' est l'ensemble des attributs communs à tous les objets de A et A' est l'ensemble des objets possédant tous les attributs de O.

Soit un contexte formel $(\mathscr{O},\mathscr{A},I)$, un *concept* formel C est un couple $\langle O,A\rangle$ où O est un sous-ensemble de $\mathscr{O}$, A un sous-ensemble de $\mathscr{A}$ tel que $O' = A$ et $A' = O$. O est alors nommé l'*extension (extent)* de C et A son *intension (intent)*. On remarque qu'il s'agit d'une opération fermée : tout objet de l'extension a tous les attributs de l'intension.

Il est possible de définir un ordre partiel sur ces concepts de la façon suivante :

$$\langle O_1,A_1\rangle \leq \langle O_2,A_2\rangle \Leftrightarrow (O_1 \subseteq O_2 \Leftrightarrow A_2 \subseteq A_1).$$

Autrement dit, un concept C_1 est dit inférieur à un concept C_2 lorsque l'extension de C_1 est un sous-ensemble strict de l'extension de C_2. L'ensemble des concepts muni de cet ordre partiel forme un treillis.

Dans ses articles, Clark définit un contexte formel $(C) = (S,C,I)$ dont les objets et attributs sont respectivement les facteurs (S) et contextes (C) présents dans l'échantillon d'apprentissage. Un couple $\langle S_i \times C_i\rangle$ appartient à la relation binaire I si $C_i \circ S_i \subseteq L$. On peut alors définir un ensemble de concepts formels grâce à ce contexte formel. Tous les facteurs d'un même concept sont considérés comme substituables dans les contextes du concept.

On note alors $B(L)$ l'ensemble des concepts du langage L. L'ordre partiel associé est tel que $\langle S_1,C_1\rangle \leq \langle S_2,C_2\rangle$ si $S1 \subseteq S2$.

De plus, dans [CE07], la concaténation de deux concepts est définie de la façon suivante : $\langle\ O_1,A_1\ \rangle \bullet \langle\ O_2,A_2\ \rangle\ =\ \langle\ (A_1.A_2)'',(A_1.A_2)'\rangle$.

Un exemple d'identification de concepts et du treillis associé à partir de mots d'un langage est donné en figure 3.3.

Génération d'une grammaire hors-contexte sous forme normale de Chomsky à partir du treillis de concepts A partir de ce treillis, il est possible de construire une grammaire $G = \langle\Sigma,N,R,S\rangle$ telle que :

- l'alphabet Σ est égal à l'ensemble des facteurs ;
- l'ensemble des non-terminaux N représente l'ensemble des concepts définis sur l'échantillon d'apprentissage ;
- l'axiome de départ S est l'ensemble des non-terminaux définis par des concepts $\langle S_i,C_i\rangle$ tel que $(\lambda,\lambda) \in C_i$;
- l'ensemble des règles de production R est défini par :
  - des règles terminales $\langle S_i,C_i\rangle \rightarrow w$ si $w \in S$
  - des règles de branchement $\langle S,C\rangle \rightarrow \langle S_1,C_1\rangle\langle S_2,C_2\rangle$ si $\langle S,C\rangle \geq \langle S_1,C_1\rangle \bullet \langle S_2,C_2\rangle$

Cette définition permet d'éviter une phase d'alignement des exemples et ainsi de récupérer l'ensemble des classes substituables présentes dans les exemples sans faire de sélection. La définition des règles préserve également toutes les ambiguïtés du langage puisque qu'elles correspondent à toutes les décompositions en non-terminaux d'un non-terminal donné. Cependant, cette simplicité formelle cache plusieurs contreparties : en

	(λ,λ)	(a,λ)	(λ,b)
λ	X		
ab	X		
a			X
b		X	

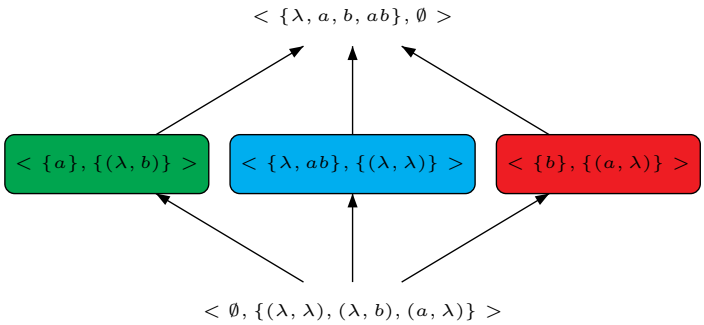


Fig. 3.3 – Identification des concepts et du treillis associés au langage $\{\lambda, ab\}$, les concepts formels sont indiqués en couleur

agissant ainsi, on obtient énormément de règles dont certaines seront redondantes. De plus, la forme normale de Chomsky nécessaire pour le calcul des règles n'est pas très explicite. Elle impose une structure très particulière aux règles qui peut être éloignée des structures originellement présentes dans les séquences. Enfin, il semble que certains concepts pourraient être dérivés à partir d'une concaténation de concepts et mériteraient un traitement particulier afin de réduire le nombre de constituants intéressants, et donc le nombre de non-terminaux, du langage appris.

3.1.4 Discussion

D'après cet état de l'art, le procédé de substituabilité semble être un bon moyen de généralisation des grammaires algébriques permettant de conserver leur expressivité ainsi que leur compacité.

Nous avons vu qu'il existe plusieurs algorithmes qui semblent efficaces afin d'inférer une grammaire à partir d'un jeu de séquences appartenant au langage voulu.

l'algorithme ABL propose un apprentissage par alignement des séquences ce qui est en parfait accord avec les méthodes d'alignements de protéines vues au premier chapitre. L'utilisation de certaines heuristiques pour choisir les alignements à conserver détermine la compacité de la grammaire apprise. Adios procède de façon plus locale. Il apprend d'abord les sous-langages les plus fréquents par extraction de motifs et génère la grammaire autour de ceux-ci, ce qui permet d'obtenir une grammaire facilement compréhensible et analysable. Cependant, les phases de sélection proposées par ces algorithmes ne permettent

pas de conserver les ambiguïtés de structure ni d'assurer l'identification à la limite du langage appris.

L'approche proposée par Clark est intéressante au niveau de la découverte et de la formalisation des classes substituables, ainsi qu'au niveau de l'apprentissage de la grammaire. Elle permet à la fois de conserver les ambiguïtés du langage et l'identification de celui-ci en tant que langage algébrique substituable, ce que nous détaillons en section suivante. Cependant la création des règles de production selon la forme normale de Chomsky rend la représentation obtenue moins lisible et très redondante. De plus, le critère utilisé pour créer des classes de facteurs substituables utilise un contexte global qui est peu adapté aux séquences protéiques, de grande taille et fortement dissimilaires sur les parties non fonctionnelles et en particulier à leurs extrémités.

3.2 Apprendre des langages substituables

Basées sur le critère de la substituabilité de Harris, les définitions récentes des classes de langages hors-contexte substituables ont conduit à des résultats intéressants d'apprenabilité polynomiale pour des langages formels expressifs. Pour mettre la théorie en pratique, nous avons conçu un algorithme d'apprentissage efficace qui construit directement depuis l'échantillon positif nécessaire une grammaire réduite canonique non redondante. Nous présentons d'abord les définitions que nous utilisons sur les langages substituables.

Nous étendons le cadre habituel en y introduisant des notions de localité et de contextualité. Ceci donne lieu à de nouvelles classes de langages que nous comparons aux classes connues. La description de l'algorithme proposé s'effectue en deux étapes : nous proposons un premier algorithme d'apprentissage avant l'introduction de notre algorithme et montrons son intérêt expérimentalement sur des ensembles de données artificiels et applicatifs.

3.2.1 Langages formels et substituabilité locale

Nous introduisons ici les définitions et notations relatives aux langages substituables.

Pour un langage L, l'ensemble de ses facteurs est $Sub(L) = \{ y \in \Sigma^* : \ x,z \in \Sigma^*, xyz \in L\}$ et l'ensemble de ses contextes est $Con(L) = \{\langle x,z\rangle \in \Sigma^* \times \Sigma^* : \ y \in \Sigma^*, xyz \in L\}$. Le contexte vide est $\langle \lambda, \lambda \rangle$.

La distribution d'un mot $y \in \Sigma^*$ dans un langage L est définie par l'ensemble de ses contextes dans L : $D_L(y) = \{\langle x,z\rangle \in \Sigma^* \times \Sigma^* : xyz \in L\}$. Deux mots y_1 et y_2 dans Σ^* sont syntaxiquement équivalents pour un langage L, noté $y_1 \equiv_L y_2$, si et seulement si $D_L(y_1) = D_L(y_2)$.

La relation d'équivalence $\equiv_L$ définit une congruence sur le monoïde Σ^* si $y_1 \equiv_L y_2$ implique $\forall x,z$ in $\Sigma^*, xy_1z \equiv_L xy_2z$. On note la classe de congruence de y par $[y]_L = \{y' \in \Sigma^* : y \equiv_L y'\}$. La classe de congruence $[\lambda]$ est appelée *classe de congruence unité*. L'ensemble $\{y : D_L(y) = \emptyset\} = \Sigma^* \backslash Sub(L)$ est appelé *classe de congruence zéro*. Une classe de congruence est non zéro si c'est un sous ensemble de $Sub(L)$. On peut noter que pour tout mot y_1, y_2 dans Σ^*, $[y_1y_2]_L \supseteq [y_1]_L[y_2]_L$.

Intéressé par l'apprenabilité des langages naturels et inspiré par l'apprentissage distributionnel, A. Clark introduit dans [CE07] les langages substituables comme une classe formelle basée sur le critère de substituabilité de Harris.

Deux mots non vides y_1 et y_2 sont dit *faiblement substituables* dans un langage L, noté $y_1 \doteq_L y_2$, s'ils apparaissent dans un même contexte, c'est à dire si et seulement si il existe $x,z \in \Sigma^*$, tel que $xy_1z \in L \wedge xy_2z \in L$. Les langages substituables sont ceux où la substituabilité faible implique la congruence syntaxique, *i.e.* tel que $y_1 \doteq_L y_2$ implique $y_1 \equiv_L y_2$ (ou, de façon équivalente, tel que $D_L(y_1) \cap D_L(y_2) \neq \emptyset$ implique $D_L(y_1) = D_L(y_2)$). Les langages substitituables se définissent donc de la façon suivante :

Définition 3.1 *[CE07] Un langage L est substituable si et seulement si pour tout $x_1,y_1,z_1,x_2,y_2,z_2 \in \Sigma^*, y_1,y_2 \neq \lambda$:*

$$x_1y_1z_1 \in L \wedge x_1y_2z_1 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L).$$

S'appuyant sur l'analogie entre la substituabilité pour les langages hors-contexte et la réversibilité introduit par [Ang82] pour les langages réguliers, R. Yoshinaka [Yos08] introduit la hiérarchie des langages hors-contexte k,l-substituables comme une contrepartie de la hiérarchie k-réversible des langages réguliers. Dans cette hiérarchie, l'expressivité augmente avec les paramètres k et l en limitant la substituabilité aux facteurs auxquels on concatène un préfixe et un suffixe de longueur k et l respectivement. Les définitions utilisées pour les langages substituables s'étendent aisément. Deux mots non vides y_1 et y_2 sont *faiblement k,l-substituables dans un contexte* $\langle u,v \rangle \in \Sigma^k \times \Sigma^l$ pour le langage L, ssi $uy_1v \doteq_L uy_2v$, soit s'il existe $x,z \in \Sigma^*$, tel que $xuy_1vz \in L \wedge xuy_2vz \in L$. Les langages k,l-substituables sont ceux tels que la substituabilité faible dans un contexte de $\Sigma^k \times \Sigma^l$ implique la congruence syntaxique dans ce même contexte, *i.e.* tel que $uy_1v \doteq_L uy_2v$ implique $uy_1v \equiv_L uy_2v$ (ou, de façon équivalente, tel que $D_L(uy_1v) \cap D_L(uy_2v) \neq \emptyset$ implique $D_L(uy_1v) = D_L(uy_2v)$) :

Définition 3.2 *[Yos08] Un langage L est k,l-substituable si et seulement si pour tout $x_1,y_1,z_1,x_2,y_2,z_2 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, uy_1v, uy_2v \neq \lambda$:*

$$x_1uy_1vz_1 \in L \wedge x_1uy_2vz_1 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L).$$

Notons que la définition de mots substituables y_1 et y_2 reste basée sur la recherche d'un contexte global commun.

Dans [CGN12], nous avons introduit pour la caractérisation d'un ensemble de protéines un critère moins strict basé sur l'utilisation d'un contexte *local* commun qui s'applique pour des cas plus pratiques. Nous introduisons deux paramètres k et l pour restreindre la longueur des contextes locaux droit et gauche pour inférer la propriété de substituabilité. Deux mots non vides y_1 et y_2 sont dit *faiblement (k,l)-localement substituables* dans un langage L, noté $y_1 \doteq_L^{k,l} y_2$, ssi il existe $x_1,x_2,z_1,z_2 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l$, tel que : $x_1uy_1vz_1 \in L \wedge$

$x_2uy_2vz_2 \in L$. Les langages k,l-localement substituables sont ceux tels que la substituabilité (k,l)-locale faible implique la congruence syntaxique, *i.e.* tel que $y_1 \doteq_L^{k,l} y_2$ implique $y_1 \equiv_L y_2$ (ou, de façon équivalente, si nous définissons $D_L^{k,l}(y)$ comme $\{\langle u,v\rangle \in \Sigma^k \times \Sigma^l : uyv \in Sub(L)\}$, tel que $D_L^{k,l}(y_1) \cap D_L^{k,l}(y_2) \neq \emptyset$ implique $D_L(y_1) = D_L(y_2)$) :

Définition 3.3 *[CGN12] Un langage L est k,l-localement substituable, noté k,l-LS, ssi pour tout*
$x_1,y_1,z_1,x_2,y_2,z_2,x_3,z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, uy_1v, uy_2v \neq \lambda$:

$$x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L).$$

En superposant la restriction locale des contextes avec celle de mots substituables des langages k,l-substituables [Yos08], nous introduisons la définition des langages k,l-localement et contextuellement substituables, que nous notons k,l-LCS :

Définition 3.4 *[CGN12] Un langage L est k,l-local contextuellement substituable ssi pour tout* $x_1,y_1,z_1,x_2,y_2,z_2,x_3,z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, uy_1v, uy_2v \neq \lambda$:

$$x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L).$$

3.2.2 Comparaison des classes de langage substituables

La classe des langages substituables définie par [CE07] se tient à la limite de la hiérarchie des langages k,l-localement substituables quand k et l tendent vers l'infini, et est la première classe dans la hiérarchie des langages k,l-substituables, avec k et l égaux à 0. Comme pour les langages réversibles, nous allons donc utiliser le terme *langage zéro substituable* pour désigner spécifiquement cette classe et utiliser *langages substituables* comme un terme générique pour les classes de langages définies grâce au critère de substituabilité.

Formellement, les langages i,j-localement et k,l-contextuellement substituables, que nous notons langages i,j-k,l-LCS, sont définis d'après les longueurs de deux types de contextes à gauche et à droite de l'implication par [1] :

1. $k \leq i \wedge l \leq j$

   pour tout $x_1,y_1,z_1,x_2,y_2,z_2,x_3,z_3 \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l, a \in \Sigma^{i-k}, b \in \Sigma^{j-l}$
   tels que $uy_1v, uy_2v \neq \lambda$,

   $$x_1auy_1vbz_1 \in L \wedge x_3auy_2vbz_3 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L)$$

2. $k \geq i \wedge l \geq j$

   pour tout $x_1,y_1,z_1,x_2,y_2,z_2,x_3,z_3 \in \Sigma^*, c \in \Sigma^{k-i}, d \in \Sigma^{l-i}, r \in \Sigma^i, s \in \Sigma^j$
   tels que $ry_1s, ry_2s \neq \lambda$,

   $$x_1cry_1sdz_1 \in L \wedge x_3ry_2sz_3 \in L \Rightarrow (x_2cry_1sdz_2 \in L \Leftrightarrow x_2cry_2sdz_2 \in L)$$

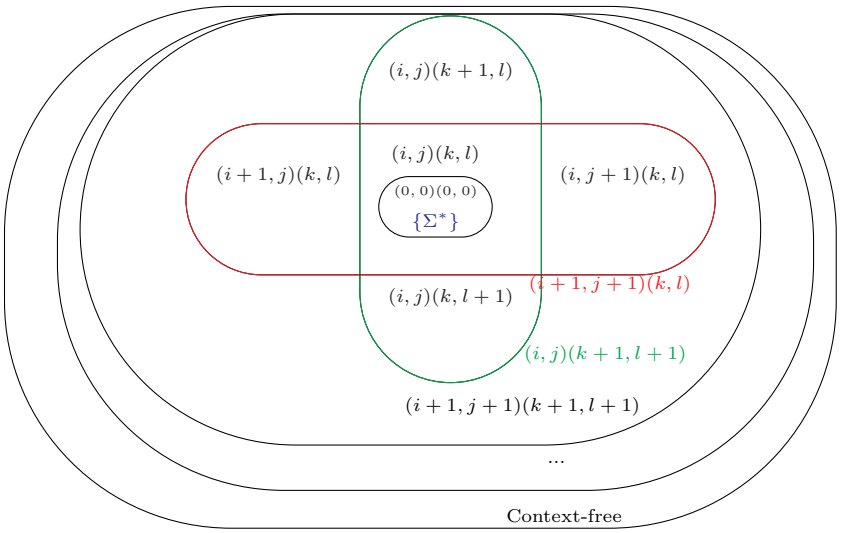


Fig. 3.4 – Hiérarchie des langages i,j-localement et k,l-contextuellement substituables

Remarquons que si un langage est i,j-k,l-LCS, alors il est aussi i,j-a,b-LCS avec $a \geq k$ et $b \geq l$ et que cette hiérarchie selon la taille des contextes est stricte. De la même façon, il est m,n-localement et k,l-contextuellement substituable avec $m \geq i$ et $n \geq j$ et la hiérarchie selon la taille des contextes locaux est stricte. Evidemment, le langage est aussi m,n-a,b-LCS pour $a \geq k$, $b \geq l$, $m \geq i$ et $n \geq j$.

La figure 3.4 montre l'inclusion des langages i,j-k,l-LCS selon les différents paramètres, la hiérarchie ayant la classe des langages hors-contexte comme borne supérieure.

La table 3.1 résume les différentes classes de langages présentées jusqu'ici dans le cadre général des langages i,j-localement et k,l-contextuellement substituables.

L'ensemble des langages k,l-LCS forme une classe attractive mixant les substitutions locales et contextuelles de manière symétrique avec peu de paramètres. De plus, on peut établir un lien entre cette classe de langage et la famille des langages localement testables, une intéressante sous-classe des langages réguliers apprenable à partir d'exemples positifs seulement, et ceci autant du point de vue théorique que pratique ([GVO90, GV90, YIK94]).

Définition 3.5 (langage strictement k-testable) *Soit $L_k(w)$ et $R_k(w)$ les préfixes et suffixes de w de taille k, $I_k(w)$ l'ensemble des facteurs non vides intérieurs de w de*

[1] On ne détaille ici que les deux cas principaux, les deux autres peuvent être déduits de manière similaire
[2] pourrait être nommé k,l-contextuellement substituable

Langage	définition locale	application contextuelle
substituable [CE07]	(∞,∞)	$(0,0)$
k,l-substituable[2] [Yos08]	(∞,∞)	(k,l)
k,l-localement substituable	(k,l)	$(0,0)$
k,l-localement et contextuellement substituable	(k,l)	(k,l)
i,j-localement et k,l-contextuellement substituable	(i,j)	(k,l)

Tab. 3.1 – Les différentes classes de langages substituables vues comme des sous-classes des langages i,j-localement et k,l-contextuellement substituables

longueur k. Un langage L sur S est strictement k-testable ssi il existe des ensembles finis A,B,C tels que $A,B,C \subseteq S^k$, et pour tout w avec $|w| \geq k$, $w \in L$ ssi $L_k(w) \in A, R_k(w) \in B, I_k(w) \subseteq C$.

Pour tout langage k-testable, on peut démontrer la propriété suivante :

$$\forall x_1,y_1,x_2,y_2 \in \Sigma^*, u \in \Sigma^k,\ x_1uy_1 \in L \wedge x_2uy_2 \in L \Rightarrow x_1uy_2 \in L \wedge x_2uy_1 \in L$$

De la même manière :

$$\forall x_2,y_1,x_3,y_2 \in \Sigma^*, u \in \Sigma^k, x_2uy_1 \in L \wedge x_3uy_2 \in L \Rightarrow x_2uy_2 \in L$$

La combinaison de ces deux propriétés conduit à l'assertion suivante :

$$\forall x_1,y_1,x_2,y_2,x_3 \in \Sigma^*, u \in \Sigma^k, x_1uy_1 \in L \wedge x_3uy_2 \in L \Rightarrow (x_2uy_1 \in L \Leftrightarrow x_2uy_2 \in L)$$

Au vu de la définition 3.4, cette dernière propriété est exactement celle des langages k,∞-LCS. Les langages k-testables sont donc k,∞-LCS. On peut procéder par symétrie et en déduire l'inclusion des langages l-testables dans les langages ∞,l-LCS.

La classe des langages k,l-LCS peut être vue comme une extension bidirectionnelle des k-testables, de la même manière que les langages k,l-substituables sont la contrepartie des langages k-réversibles. La figure 3.5 montre un aperçu de ces différentes classes de langages et de leurs liens.

3.2.3 Propriétés de clôture

[Yos08] a démontré un certain nombre de propriétés sur les langages k,l-substituables. Nous présentons ici des résultats similaires sur les langages localement substituables.

Proposition 1 *Les langages k,l-localement substituables ne sont pas clos par intersection avec des ensembles réguliers*

Soient $L_0 = ae^*cf^*a \cup ae^*df^*a \cup be^*cf^*b$ et $L_1 = L_0 \cup be^*df^*b$. L_0 est régulier et L_1 est $1,1$-localement substituable. Mais $L_1 \cap L_0 = L_0$ n'est pas substituable quelque soit k, l. Exemple : $ae^kcf^la \in L_0 \wedge ae^kdf^la \in L_0 \not\Rightarrow (be^kcf^lb \in L_0 \Leftrightarrow be^kdf^lb \in L_0)$

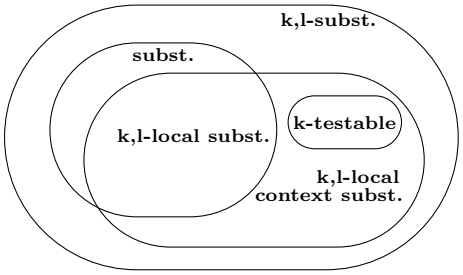


Fig. 3.5 – Inclusion des différentes classes de langages substituables et k-testables

Proposition 2 *Les langages k,l-localement substituables ne sont pas clos par union*

Soient $L_2 = ae^*cf^*a \cup ae^*df^*a$ et $L_3 = be^*cf^*b$. L_2 est $1,1$-localement substituable. Mais $L_2 \cup L_3 = L_0$ n'est pas k,l-substituable quelque soit k, l.

Proposition 3 *Les langages k,l-localement substituables ne sont pas clos par concaténation*

Soient $L_4 = ae^*c$ et $L_5 = e^*a$. L_4 et L_5 sont k,l-localement substituable. Mais $L_4L_5 = ae^*ce^*a$ n'est pas k,l-localement substituable. Exemple : $ae^kee^lcea \in L_4L_5 \wedge ae^kce^la \in L_4 \cap L_5 \nRightarrow ae^xe^kee^lce^xa \in L_4L_5 \Leftrightarrow ae^xe^kce^lce^xa \in L_4L_5$

Proposition 4 *Les langages k,l-localement substituables ne sont pas clos par complémentaire*

$L_6 = a^*b$ est k,l-localement substituable, mais le complémentaire L_6^c n'est pas k,l-localement substituable quelque soit k, l. Exemple : $ba^kaa^l \in L_6^c \wedge ba^kba^l \in L_6^c \nRightarrow a^kba^lb \in L_6^c \Leftrightarrow a^kaa^lb \in L_6^c$

Proposition 5 *Les langages k,l-localement substituables ne sont pas clos par clôture de Kleene*

$L_7 = (a^*ba^*)^+d$ est k,l-localement substituable, mais la clôture de kleene n'est pas k,l-localement substituable quelque soit k,l. Exemple : $a^kba^ld \in L_7^* \wedge aba^kda^lbad \in L_7^* \nRightarrow a^kba^ld \in L_7^* \Leftrightarrow a^kda^ld \in L_7^*$

Proposition 6 *Les langages k,l-localement substituables ne sont pas clos par homomorphisme non vide*

$L_8 = ae^*cf^*a \cup be^*cf^*b \cup gy^*dx^*g$ est k,l-localement substituable. Soit h est un homomorphisme : $h(a) = a, h(b) = b, h(c) = c, h(d) = d, h(e) = e, h(f) = f, h(g) = a, h(x) = f, h(y) = e$. $h(L_8) = L_0$ n'est pas k,l- localement substituable.

Proposition 7 *Les langages k,l-localement substituables ne sont pas clos par homomorphisme inverse*

$L_6 = a^*b$ est k,l-localement substituable. Soit h tel que $h(a) = a, h(b) = b, h(e) = \Lambda$. $h^{-1}(L_6)$ n'est pas k,l-localement substituable. Exemple : $e^k ae^l b \in h^{-1}(L_6) \wedge e^k ae^l b \in h^{-1}(L_6) \nRightarrow be^k ee^l \in h^{-1}(L_6) \Leftrightarrow be^k ae^l \in h^{-1}(L_6)$

Proposition 8 *Les langages k,l-localement substituables ne sont pas clos par renversement (pour les langages k,l-localement substituables avec $k \neq l$)*

Si L est k,l-localement substituable, son langage miroir est l,k-localement substituable. Donc, si $k \neq l$, les deux langages n'appartiennent pas à la même classe.

Proposition 9 *Les langages k,l-localement substituables sont clos par intersection*

Soient L et L' k,l-localement substituables. Si $x_1vy_1uz_1$,
$x_3vy_2uz_3, x_2y_1z_2 \in L \cap L'$ pour $v \in \Sigma^k, u \in \Sigma^l$, alors ils sont à la fois dans L et L'. Etant donné que L et L' sont k,l-localement substituables, $x_2y_2z_2$ est à la fois dans L et L' et ainsi dans $L \cap L'$.

Proposition 10 *Les langages k,l-localement substituables sont clos par homomorphisme inverse non vide*

Soit L un langage k,l-localement substituable et h un homomorphisme non vide. Notons $h(w) = \overline{w}$ pour la lisibilité. Si $x_1uy_1vz_1, x_3uy_2vz_3$,
$x_2uy_1vz_2 \in h^{-1}(L)$ pour $u \in \Sigma^k, v \in \Sigma^l$ et $uy_1v, uy_2v \in \Sigma^+$, alors $\overline{x_1uy_1vz_1}, \overline{x_1uy_2vz_1}, \overline{x_2uy_1vz_2} \in L$. Étant donné que L est k,l-localement substituable et $|\overline{u}| \geqslant |u| = k$, $|\overline{v}| \geqslant |v| = l$, $|\overline{uy_1v}|, |\overline{uy_2v}| \geqslant 1$, nous avons $\overline{x_2uy_2vz_2} \in L$ et donc $x_2uy_2vz_2 \in h^{-1}(L)$.

La plupart des résultats sont négatifs, sauf la clôture par homomorphisme inverse non vide et par intersection. Cette dernière proposition suggère, comme pour les langages locaux, que des méthodes d'inférence grammaticales incluant des connaissances expertes en renommant les symboles dans les séquences [GVC87] pourraient être développées pour apprendre des langages localement substituables.

3.2.4 La substituabilité comme principe de généralisation

Nous considérons le problème de l'apprentissage des langages substituables grâce à des algorithmes utilisant des représentations de grammaire hors-contexte. Et pour cela nous utilisons le principe de généralisation suivant :

$$y_1 \doteq_L y_2 \implies y_1 \equiv_L y_2.$$

Autrement dit, une substituabilité faible découverte dans l'échantillon d'apprentissage sera généralisée en une classe de congruence syntaxique. Pour chaque classe de langage définie, on peut associer une version spécifique de ce principe de généralisation.

Pour découvrir les substituabilités à généraliser, on partitionne les facteurs de l'échantillon d'apprentissage en classes de substituabilité. A chaque classe sera associé un nouveau non-terminal et différentes productions correspondant aux différents mots de la classe.

Étant donné un ensemble d'exemples positifs, les langages qui vont être appris selon les différents critères de substituabilité ne sont pas indépendants. Avant d'étudier ces relations, nous commençons par introduire un petit exemple illustratif de cette notion.

On utilise un même ensemble de données $K = \{abcde, abfde, yzcji, vzmjk\}$ pour les différentes classes de langages cibles pour des petites valeurs de k et l. L'ensemble des règles R de la grammaire $G = \langle \Sigma, N, R, S \rangle$ ainsi que le langage L attendus pour chaque principe de généralisation sont donnés. Pour chacune des grammaires, on a $\Sigma = \{a, b, c, d, e, f, i, jk, m, v, yz\}$ et $N = \{S, X_i\}$.

- substituabilité ([CE07])
  $S \rightarrow abXde \mid yzXji \mid vzmjk$
  $X \rightarrow c \mid f$
  $L = \{abcde, abfde, yzcji, vzmjk, yzfji\}$
- substituabilité $1,1$-contextuelle ([Yos08])
  $S \rightarrow aXe \mid yzcji \mid vzmjk$
  $X \rightarrow bcd \mid bfd$
  $L = \{abcde, abfde, yzcji, vzmjk\}$
- substituabilité $1,1$-locale
  $S \rightarrow abXde \mid yzXji \mid vzXjk$
  $X \rightarrow c \mid f \mid m$
  $L = \{abcde, abfde, yzcji, vzmjk, yzmji, vzcjk, yzfji, vzfjk, abmde\}$
- substituabilité $1,1$-locale et contextuelle
  $S \rightarrow aXe \mid yX_2i \mid vX_2k$
  $X \rightarrow bcd \mid bfd$
  $X_2 \rightarrow zmj \mid zcj$
  $L = \{abcde, abfde, yzcji, vzmjk, yzmji, vzcjk\}$

Plus généralement, étant donné un ensemble d'apprentissage K, on peut établir une hiérarchie entre les différents langages appris selon les contraintes de généralisation. Si $L_X(K)$ dénote le langage cible X incluant K, on peut démontrer les propriétés suivantes :

Proposition 1 $L_{k,l\text{-}substitutable}(K) \subseteq L_{k,l\text{-}LCS}(K)$

Preuve: $x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L)$
Si $x_1 = x_3 \wedge z_1 = z_3$:
$x_1uy_1vz_1 \in L \wedge x_1uy_2vz_1 \in L \Rightarrow (x_2uy_1vz_2 \in L \Leftrightarrow x_2uy_2vz_2 \in L)$
Donc, tous les mots ajoutés pour satisfaire la substituabilité k,l-contextuelle sont aussi ajoutés pour la substituabilité k,l-locale et contextuelle .

Proposition 2 $L_{substitutable}(K) \subseteq L_{k,l\text{-}LS}(K)$

Preuve: $x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L)$
Si $x_1u = x_3u(= x_4) \wedge vz_1 = vz_3(= z_4)$:
$x_4y_1z_4 \in L \wedge x_4y_2z_4 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L)$

Proposition 3 $L_{k,l\text{-}LCS}(K) \subseteq L_{k,l\text{-}LS}(K)$

Preuve: $x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_2y_1z_2 \in L \Leftrightarrow x_2y_2z_2 \in L)$
Si $x_2 = x_4u \wedge z_2 = vz_4$:
$x_1uy_1vz_1 \in L \wedge x_3uy_2vz_3 \in L \Rightarrow (x_4uy_1vz_4 \in L \Leftrightarrow x_4uy_2vz_4 \in L)$

La hiérarchie définie par le principe de généralisation découle de la hiérarchie entre les classes de langages étudiée précédemment.

3.2.5 Un premier algorithme générique d'apprentissage pour les langages substituables

Afin d'apprendre des classes de langages substituables, [CE07] et [Yos08] ont introduit plusieurs algorithmes pour la classe des substituables et k,l-substituables.

Dans un premier temps, nous avons travaillé sur une extension de ces algorithmes à la classe des langages localement substituables, travail qui a fait l'objet d'une publication dans [CGN12], où nous avons implémenté l'algorithme 1 qui est une adaptation de l'algorithme SGL de [CE07].

Étant donné un ensemble de séquences K et les paramètres k et l spécifiant la classe cible des langages substituables souhaitée, l'algorithme partitionne tous les facteurs de K en classes de substituabilité en utilisant un graphe de substitution et retourne une grammaire hors-contexte en forme normale de Chomsky. Cette grammaire contient un *symbole non-terminal* $[\![C]\!]$ pour chaque classe de substituabilité C. Des *règles de branchement* $[\![C_K(y_1y_2)]\!] \rightarrow [\![C_K(y_1)]\!][\![C_K(y_2)]\!]$ — où $C_K(x)$ identifie la classe de substituabilité de x — sont créées pour chaque concaténation possible y_1y_2. Ces règles permettent la généralisation dans le sens où chaque facteur de l'échantillon d'apprentissage peut maintenant être remplacé par sa classe de substituabilité. Enfin, *les règles terminales* sont créées qui génèrent les symboles terminaux.

L'algorithme présenté utilise un critère de généralisation permettant d'inférer des langages localement substituables mais on peut remarquer qu'il peut être adapté à d'autres classes de langages substituables, simplement en changeant la fonction qui retourne les classes de substituabilité. Cela peut être fait en modifiant la ligne 2 de l'algorithme 2 qui définit la relation entre les facteurs dans le graphe de substituabilité. La définition de l'ensemble des arcs doit être remplacée pour les langages substituables par : $E \leftarrow \{\{y_1,y_2\} \in V \times V : xy_1z \in K, xy_2z \in K, y_1 \neq y_2, x \in \Sigma^*, z \in \Sigma^*\}$, pour les langages k,l-substituables par : $E \leftarrow \{\{uy_1v, uy_2v\} \in V \times V : xuy_1vz \in K, xuy_2vz \in K, y_1 \neq y_2, x \in \Sigma^*, z \in \Sigma^*, u \in \Sigma^k, v \in \Sigma^l\}$ et pour les langages k,l-localement et contextuellement substituables par : $E \leftarrow \{\{uy_1v, uy_2v\} \in V \times V : uy_1v \in Sub(K), uy_2v \in Sub(K), y_1 \neq y_2, u \in \Sigma^k, v \in \Sigma^l\}$.

Même si l'algorithme ne retourne pas toujours une grammaire appartenant à la classe de langage cible, il permet d'identifier la grammaire cible en un temps polynomial à partir

Algorithm 1: SGL_{LS} (Graphe d'apprentissage des substitutions pour les langages k,l-localement substituables)

Input: Ensemble de séquences K sur l'alphabet Σ, entier k, entier l
Output: Grammaire $G = \langle \Sigma, N_K, S_K, P_K \rangle$

```
/* Partition de Sub(K) en classes de substituabilité */
```
$\mathscr{C}_K \leftarrow$ `Local_substitutability_classes`(K,k,l)
$\forall y \in Sub(K), C_K(y) = C \in \mathscr{C}_K : y \in C$
```
/* Construction de la grammaire */
```
$N_K \leftarrow \emptyset, P_K \leftarrow \emptyset$
for $C \in \mathscr{C}_K$ **do**
  ```
  /* Un non-terminal pour chaque classe de substituabilité */
  ```
  $N_K \leftarrow N_K \cup \{[\![C]\!]\}$
  if $C \cap K \neq \emptyset$ **then**
    $S_K \leftarrow [\![C]\!]$
  ```
  /* Règles de production pour chaque mot d'une classe */
  ```
  for $y \in C$ **do**
    if $|y| > 1$ **then**
      ```
      /* Règles de branchement : une règle 'CNF' pour chaque
      décomposition possible */
      ```
      for $y_1 \in \Sigma^+, y_2 \in \Sigma^+ : y_1 y_2 = y$ **do**
        $P_K \leftarrow P_K \cup \{[\![C]\!] \rightarrow [\![C_K(y_1)]\!][\![C_K(y_2)]\!]\}$
    else
      ```
      /* Règles terminales */
      ```
      $P_K \leftarrow P_K \cup \{[\![C]\!] \rightarrow y\}$
return $\langle \Sigma, N_K, S_K, P_K \rangle$

Algorithm 2: *Local_substitutability_classes*

Input: Ensemble de séquences K sur l'alphabet Σ, entier k, entier l
Output: Les classes de substituabilité k,l-locales faibles de K

```
/* Construction d'un graphe de substituabilité avec les facteurs
de K */
```
$V \leftarrow \{y \in \Sigma^+ : y \in Sub(K)\}$
$E \leftarrow \{\{y_1, y_2\} \in V \times V : uy_1v \in Sub(K), uy_2v \in Sub(K), y_1 \neq y_2, u \in \Sigma^k, v \in \Sigma^l\}$
```
/* Retourne les composantes connexes du graphe */
```
return `Connected_components`$(\langle V, E \rangle)$

d'un échantillon caractéristique de taille polynomiale en $|G|.\tau_G$, où τ_G est la taille de la grammaire, à condition que les classes de substituabilité soit connues [CE07, Yos08].

Cependant, même si l'algorithme est polynomial, il ne peut pas être utilisé en pratique

sur des jeux de données réels comme en témoignent nos premières expériences sur les protéines [CGN12]. En effet, chaque exécution de l'algorithme nécessite plusieurs heures, ce qui nous a empêché de faire des études plus poussées sur l'influence des paramètres ou simplement d'envisager un protocole de validation croisée pour estimer la pertinence des grammaires apprises.

Un facteur déterminant de la complexité d'un tel algorithme est le haut niveau d'ambiguïté et de redondance des grammaires considérées. Du point de vue de l'analyse ou même avec le simple objectif de vérifier et d'interpréter manuellement les règles, les grammaires obtenues ont une taille trop importante, un problème qui touche également le temps d'apprentissage.

Un exemple d'execution de cette algorithme est donné plus loin, en sous-section 3.3.4.

3.3 Apprentissage de grammaires réduites

3.3.1 Grammaires réduites

Pour améliorer aussi bien le temps d'analyse et la lisibilité de la grammaire apprise par SGL_{LS} que les arbres de dérivation, nous proposons d'abandonner la forme normale de Chomsky au profit d'une *forme réduite* qui évite les pas de dérivation inutiles et les redondances. Ceci suppose l'introduction de deux types d'opération :

- éliminer les non-terminaux inutiles dont la dérivation est déterministe (si un non-terminal A est la partie gauche d'une seule règle $A \rightarrow \alpha$, le non-terminal et la règle sont supprimés et chaque occurrence de A dans une autre règle est remplacée par α) ;
- minimiser les parties droites des règles de production (remplacer chaque facteur β d'une partie droite par les plus petits facteurs α dans $N \cup \Sigma$ desquels β peut être dérivé)

La première réduction correspond à supprimer les non-terminaux qui donnent lieu à des *vacuous local derivation trees* introduits par [Cla11]. Nous appelons ce type de classes de congruence *composite* car elles peuvent être factorisées par la concaténation d'un ensemble de classes de congruence. Plus formellement :

Définition 3.6 (Classe de congruence composite) *Étant donné un langage L ayant un ensemble fini de classes de congruence non zéro C^+. Une classe $[y] \in C^+$ est composite pour L ssi :*

$$\exists [x_1], \ldots [x_m] \in C^+, m \geq 2, [y] = [x_1] \ldots [x_m]$$

Une classe de congruence est *première* si elle n'est pas composite. Ces classes sont celles qui doivent être gardées comme non-terminaux. Dans l'algorithme, nous utilisons une caractérisation plus efficace des classes de congruence premières (et composites) qui peut être déduite à partir des propriétés de congruence syntaxique :

Propriété 3.1 ***classes premières*** *Étant donné un langage L ayant un ensemble fini de classe de congruence non zéro C^+. Une classe $[y]$ dans C^+ est première pour L ssi* $\forall y_1 y_2 \in [y], [y] \not\subset [y_1][y_2]$.

Preuve: Il s'agit d'une simple application du fait que les contextes d'une chaîne comprennent la concaténation des facteurs de ses contextes. Toute équation impliquant $m > 2$ facteurs est alors remplacée par une équation impliquant seulement le premier facteur et la concaténation des autres facteurs. De plus, l'équation peut être remplacée par une inclusion étant donné que l'autre direction est déjà garantie.

La seconde réduction, comme les non-terminaux et les langages qu'ils génèrent sont fixés, peut être vue comme une extension du *minimal grammar parsing* [CCGL11]. Chaque non-terminal génère un ensemble de mots substituables qui peuvent se décomposer de plusieurs façons avec les non-terminaux générés à l'étape précédente. Le problème est de trouver les différentes analyses possibles de chaque partie droite de règles de façon à conserver celles qui soient les plus simples (ici, simple signifie minimale en termes de taille de règles et non redondantes). Cela peut être résolu de façon similaire par programmation dynamique sur des graphes représentant les parties droites des règles de productions comme des mots, avec la difficulté ajoutée que tous les chemins non redondants doivent être trouvés. Résoudre ce problème est équivalent à chercher pour chaque partie droite α l'ensemble des mots non redondants générant toutes les séquences générées depuis α (voir la prochaine sous-section pour les détails). On notera que l'ensemble de règles retourné par cet algorithme est le même que celui retourné par l'algorithme récemment proposé dans [Cla14] même si les approches diffèrent : construire directement l'ensemble des règles non redondantes par programmation dynamique dans notre cas, contre filtrer dans l'ensemble complet de règles potentiellement valides celles nommées pléonastiques, qui correspondent à des règles non redondantes. Basé sur le « lemme fondamental » (propriété 3.2) des langages substituables prouvé dans [Cla14], la *forme réduite* calculé par cet algorithme est alors une *forme canonique* de grammaire pour les langages substituables.

Propriété 3.2 ***Unicité de la décomposition en classes premières*** *Toute classe de congruence non unité et non zéro d'un langage substituable L possède une décomposition unique en classes premières.*

Jusqu'à présent, nous avons seulement discuté de la réduction de la grammaire dans une forme minimale canonique non redondante. Nous présentons dans la sous-section suivante l'algorithme ReGLiS basé sur ces idées de programmation dynamique, mais en apprenant cette forme réduite en même temps que le contenu des classes de substituabilité par une réduction ascendante de l'échantillon.

3.3.2 Apprentissage d'une grammaire réduite : l'algorithme ReGLiS

L'algorithme ReGLiS est détaillé dans l'algorithme 3. En prenant comme entrée un échantillon de mots K sur un alphabet Σ, et deux paramètres k et l définissant la classe

cible de langage localement substituable, ReGLiS retourne une grammaire de forme réduite du langage cible quand K comprend un échantillon caractéristique du langage pour SGL_{LS}.

Comme SGL_{LS} présenté dans la section 3.2.5, ReGLiS construit une grammaire basée sur un ensemble de classes d'équivalence $\mathscr{C}_K$ définies sur l'évidence d'une substituabilité faible sur K. On peut remarquer que le reste de l'algorithme est indépendant de la classe du langage cible : les classes sont basées ici sur une substituabilité k,l-locale, mais d'autres critères de substituabilité faible peuvent être utilisés pour adapter l'algorithme à d'autres classes de langages substituables.

Contrairement à SGL_{LS} qui introduisait un non-terminal pour chaque classe d'équivalence, ReGLiS n'introduit un non-terminal que si la classe de substituabilité contient plus d'un mot et satisfait la propriété 3.1 sur $\mathscr{C}_K$. Ces classes sont appelées des classes de substituabilité K-premières. Le symbole introduit pour la classe de substituabilité K-première comprenant K est l'axiome de départ. Pour chaque non-terminal introduit, la grammaire contient des règles permettant de dériver en une étape chaque mot de la classe de substituabilité associée. Jusque là, la grammaire construite est seulement capable de générer les mots de K et contient des règles inatteignables joignant chaque non-terminal avec les facteurs de sa classe de substituabilité.

L'étape de généralisation au cœur de l'algorithme est obtenue par un processus de réécriture ascendant des parties droites des règles de production, en commençant par la plus petite et en optimisant l'ensemble des règles de branchement par programmation dynamique sur un graphe d'analyse pour chaque partie droite.

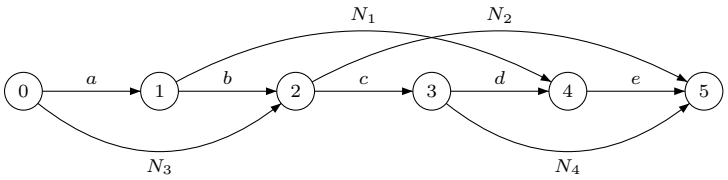


Fig. 3.6 – Exemple de graphe d'analyse pour $abcde$: les parties droites non redondantes sont aN_1e et N_3N_2.

Plus précisément, pour chaque partie droite α, un *graphe d'analyse* est construit par l'algorithme 4 (voir l'exemple de la figure 3.6). Intuitivement, le graphe représente toutes les générations possible de facteurs à partir de α par les non-terminaux disponibles. Trouver toutes les parties droites non redondantes les plus générales permettant de produire α est une tâche décrite par l'algorithme 5. En oubliant les symboles, cet algorithme se concentre sur les chemins dans le graphe d'analyse. Les chemins sont représentés par des séquences de nœuds, et ajouter un nœud i à la fin d'un chemin π est noté par la concaténation $\pi.i$. Un chemin $\pi_2 = t_1..t_m$ est réductible en $\pi = s_1..s_n$, noté $\pi \prec_r \pi_2$, si π est une sous-séquence stricte de π_2, $s_1 = t_1$ et $s_n = t_m$. Un chemin π est dit irréductible s'il est minimal pour l'ordre partiel $\prec_r$. L'algorithme 5 implémente une recherche par programmation dynamique de tous les chemins irréductibles dans le graphe d'analyse et retourne alors toutes les parties

droites non redondantes qui généralisent la partie droite α qui peuvent être utilisées pour la remplacer.

Deux améliorations ont été incluses dans cet algorithme pour permettre l'identification du langage cible pour certains échantillons sur lesquels SGL_{LS} n'avait pas réussi. Premièrement, si durant la réduction une partie droite est déjà existante dans l'ensemble de règles, le non-terminal correspondant est unifié pour éviter de générer les mêmes facteurs avec deux non-terminaux différents et toujours assurer la transitivité des classes de substitution (un mot n'appartient qu'à une classe de substituabilité à la fois). Deuxièmement, comme les langages reconnus par les non-terminaux augmentent, de nouveaux arcs peuvent apparaître dans les graphes d'analyse. Dans de tels cas, la recherche de règles minimales et non redondantes doit être répétée. La dernière boucle de l'algorithme assure la convergence vers un ensemble de règles non redondantes et requiert alors moins d'exemples pour construire la grammaire que SGL_{LS}.

Un exemple de grammaire obtenue grâce à cet algorithme est montré en sous-section 3.3.4.

3.3.3 Complexité de l'algorithme ReGLiS

Étant donné K, un ensemble d'apprentissage de taille $|K|$, et n la taille de sa séquence la plus longue, la complexité de l'algorithme 3 est déterminée par trois points critiques :

ligne 3 La création des classes de substituabilité locales, qui dépend du nombre de facteurs dans $Sub(K)$. Ce nombre est un paramètre important pour le reste de l'algorithme car toutes les opérations sont effectuées sur les éléments de $Sub(K)$.

ligne 9 La réduction du nombre de classes, qui réduit le nombre de facteurs à considérer dans la partie suivante de l'algorithme.

ligne 16 La minimisation de la grammaire, qui inclut la création des graphes d'analyse apparaît comme la plus grande source de complexité. Il est facile de voir que la complexité de cette partie dépend du nombre de règles de la grammaire.

Création des classes de substituabilités locales

La complexité de $Local\_substitutability\_classes$ dépend du nombre des éléments dans $Sub(K)$. En effet, l'algorithme commence par construire le graphe de substituabilité (graphe des contextes partagés) en créant un nœud pour chaque préfixe à chaque position de chaque séquence (i.e. création de l'ensemble des facteurs de $Sub(K)$). Pendant cette boucle, une table T est créée qui donne pour chaque contexte, l'ensemble des facteurs associés.

Pour une séquence de taille m, le nombre maximal de facteurs créés dans $\mathscr{C}_K$ et dans T est $\frac{m(m+1)}{2}$. Comme n est la taille de la plus longue séquence dans K, le nombre total de facteurs est $O(|K|n^2)$.

L'objectif est ensuite de trouver les composantes connexes maximales du graphe de substituabilité. Par définition, chaque entrée de la table T pointe vers une composante connexe. Au lieu de créer l'ensemble du graphe de substituabilité, il est suffisant de créer

Algorithm 3: ReGLiS (Learning **R**educed **G**rammar by k,l-**L**ocal **S**ubstitutability)

Input: Ensemble de séquences K sur l'alphabet Σ, entier k, entier l
Output: Grammaire $G = \langle \Sigma, N_K, S_K, P_K \rangle$

```
   /* Partition de Sub(K) en classes de substituabilité */
1  C_K ← Local_substitutability_classes(K,k,l)
   /* Non-terminaux pour pour chaque classe première et leur règles
   de production associés pour chacun de leur facteur */
2  N_K ← ∅, P_K ← ∅
3  for C ∈ C_K do
       /* Axiome */
4      if C ∩ K ≠ ∅ then
5          N_K ← N_K ∪ {[[C]]}
6          S_K ← [[C]]
7          for y ∈ C do
8              P_K ← P_K ∪ {[[C]] → y}
       /* Non-terminal correspondant aux classes de substituabilité
       premières */
9      else if (|C| > 1) and (∄C' ∈ C_K : ∀y ∈ C, ∃y' ∈ C', ∃v ∈ Σ+, y = y'v) and
       (∄C' ∈ C_K : ∀y ∈ C, ∃y' ∈ C', ∃u ∈ Σ+, y = uy') then
10         N_K ← N_K ∪ {[[C]]}
11         for y ∈ C do
12             P_K ← P_K ∪ {[[C]] → y}
   /* Généralisation */
13 repeat
14     P ← P_K ; P_K ← ∅
       /* Règles de branchement */
15     for ([[C]] → α) ∈ P ordered by increasing |α| do
16         PG ← Build_parsing_graph(α,P)
17         for β ∈ Non_redundant_rhs(PG) do
18             if ∃([[C']] → β) ∈ P_k, [[C']] ≠ [[C]] then
19                 Unify([[C']],[[C]],P_K)
20             P_K ← P_K ∪ ([[C]] → β)
21 until P_K = P
22 return ⟨Σ, N_K, S_K, P_K⟩
```

une chaîne pour tous les ensembles de facteurs de la table T. Le graphe résultat contient les même composantes connexes que le graphe entier et la complexité reste en $\mathbf{O}(|\mathbf{K}|\mathbf{n}^{\mathbf{2}})$.

Il est à noter que la classe $C_K(x)$ de chaque facteur x peut être stockée pendant la phase de recherche de composante connexe.

Algorithm 4: *Build_parsing_graph*

Input: Séquence α, Ensemble de règles P
Output: Graphe d'analyse $\langle V, E \rangle$

```
1 V ← {i ∈ [0, |α|]} /* nœuds */
2 E ← ∅ /* arcs dirigés étiquetés */
3 for i ∈ V do
4     for j ∈ V : i < j and (i, j) ≠ (O, |α|) do
5         if ∃([[C]] → α[i..j]) ∈ P then
6             E ← E ∪ (i, j, [[C]])
7 return ⟨V, E⟩
```

Algorithm 5: *Non_redundant_rhs*

Input: Graphe d'analyse : $\langle V, E \rangle$
Output: Ensemble de parties droites de règles de production non redondantes déduites du graphe d'analyse : R

```
1  R ← ∅
2  paths[0] ← {{0}}
3  for i ← 1 to |V| do
       /* mémorise l'ensemble des chemins irréductibles arrivant en i
       */
4      P ← ⋃_{(j,i,l)∈E} (paths[j].i)
5      paths[i] ← {x ∈ P : ∄y ∈ P, y ≺_r x}
6  for path ∈ paths[n] do
7      rhs ← λ
8      for i ← 1 to |path| do
9          rhs ← rhs.β_i avec β_i : (path[i − 1], path[i], β_i) ∈ E
10     R ← R ∪ rhs
11 return R
```

Réduction du nombre de classes Dans la boucle ligne 3-12, chaque élément (composante connexe) de $\mathscr{C}_K$ est visité une fois, les boucles intérieures sont exécutées $O(|K|n^2)$ fois.

Globalement, la ligne 9 visite tous les facteurs ($O(|K|n^2)$) une fois, et pour chaque facteur, tous ses préfixes et suffixes ($O(2n)$). Chaque facteur est visité seulement une fois globalement car il n'apparaît que dans une seule composante connexe par définition. Le traitement de chaque facteur de chaque classe implique une intersection de tous ses préfixes avec les autres préfixes des autres facteurs de la même classe, une opération linéaire selon le nombre de préfixes, i.e. en $O(n)$. Donc, la complexité globale de cette boucle est $\mathbf{O(n^3)}$.

En pratique cette boucle réduit le nombre de classes et le nombre de facteurs qui appartiennent à une classe de substituabilité. Le nouvel ensemble de facteurs disponibles est alors noté $K - Premier$.

Généralisation et minimisation de la grammaire Dans cette partie, (ligne 13 à 21), la boucle principale est répétée jusque qu'à ce qu'il n'y ait plus de partie droite α qui puisse être réduite. Le pire cas pour cette boucle apparaît quand tous les facteurs de taille $\leq l$, $l = 1..n$ sont considérés tour à tour pour la réduction de α. Ainsi, cette boucle est répétée au pire n fois.

La complexité de la boucle intérieure (ligne 15 à 20) est bornée par le nombre de parties droites de règles de la grammaire $\mathscr{A} = \{\alpha \mid X \to \alpha \in P_K\}$ et leur longueur puisque la boucle considère tous les α pour les graphes d'analyse.

Initialement, le nombre de règles est borné par $|K - Premier|$ et la longueur de α est $O(n)$. donc le premier tour de boucle est borné en $\mathbf{O(|K - Premier|n)}$.

Le nombre de règles peut augmenter à chaque tour de boucle, mais si cela arrive, la longueur des nouvelles règles décroît.

Si aucun facteur n'est enlevé durant la première étape de l'algorithme ($|K - Premier| = |Sub(K)|$), le pire cas est alors d'obtenir $|\mathscr{A}| = 2^{n-1}$ règles pour la grammaire cible (forme normale de Chomsky). Cela arrive quand chaque règle peut être coupée en deux pour chaque position de α (tous les facteurs de α sont factorisables par un non-terminal existant), donnant ainsi le nombre maximal de chemin irréductible dans un graphe d'analyse.

Si certains facteurs sont enlevés, le nombre de chemins irréductibles décroît forcément. Pour chaque séquence s_i de K, il existe une décomposition $s_i = u_1 v_1 u_2 v_2 ... u_p$ où $u_j, j \in 1..p$ est un facteur appartenant à une classe première (facteur premier), $v_j, j \in 1..p-1$ est un facteur qui n'est pas premier. Nous avons vu que pour toute classe non première, il existe une décomposition unique en classes premières. Ainsi, pour chaque décomposition de s_i, si u_1 est fixé, $v_1 u_2 v_2 ... u_p$ possède une décomposition unique (soit un non-terminal s'il s'agit d'un facteur de $K - Premier$, soit une décomposition en classes premières sinon). Le nombre de décompositions d'une séquence est alors borné par le nombre de préfixes appartenant à $K - premier$ applicable. Pour chaque facteur de $K - premier$, il y a donc au plus $|K - premier|$ décompositions, soit $|K - premier|^2$ règles de productions dans la grammaire finale.

Ainsi, la complexité de cette partie atteint $\mathbf{O(|K - Premier|^2 n^2)}$.

La complexité globale est donc $\mathbf{O(max(n^3, |K - Premier|^2 n^2))}$. Dans le pire cas, $|K - Premier|$ est égal au nombre de facteurs présents dans l'échantillon d'apprentissage et la complexité globale est alors de $O((|K|n^2)^2 n^2)$, soit $O(n^6)$.

3.3.4 Exemple de réduction d'une grammaire pour le langage naturel

Nous montrons ici l'intérêt des classes premières et de la forme de grammaire réduite sur un exemple simple de langage naturel.

Soit un ensemble de séquences d'apprentissage K :

$$\mathbf{K} = \{\text{"Mr Smith was here yesterday morning."},$$
"Mr Smith went here yesterday morning.",
"Mr Smith will be there tomorrow morning.",
"He will be gone tomorrow evening."$\}$

.

La grammaire suivante a été obtenue avec l'algorithme SGL_{LS}, sans avoir choisi les classes premières comme non-terminaux et en gardant la forme normale de Chomsky pour décomposer les règles :

$X_{47} \rightarrow' yesterday'$
$X_{46} \rightarrow X_3X_{43}|X_{11}X_{19}|X_{23}X_{13}$
$X_{45} \rightarrow X_{20}X_2$
$X_{44} \rightarrow X_{20}X_1|X_{45}X_{29}|X_{34}X_{16}|X_9X_{15}$
$X_{43} \rightarrow X_{20}X_{19}|X_{45}X_{13}$
$X_{42} \rightarrow' tomorrow'$
$X_{41} \rightarrow X_{42}X_{15}$
$X_{40} \rightarrow X_{30}X_{46}|X_{21}X_{43}|X_{39}X_{19}|X_{25}X_{13}|X_{21}X_{34}|X_8X_{13}$
$N_0 \rightarrow X_{30}X_{24}|X_{21}X_{31}|X_{10}X_{32}|X_{36}X_{17}|X_{26}X_{15}|X_{39}X_1$
$|X_{25}X_{29}|X_{40}X_{41}|X_{21}X_{44}|X_8X_{29}|X_{40}X_{16}|X_{33}X_{15}$
$X_{29} \rightarrow X_{13}X_{16}|X_4X_{15}|X_{13}X_{41}|X_{38}X_{15}$
$X_{28} \rightarrow X_{27}X_{47}$
$X_{25} \rightarrow X_{30}X_{23}|X_{21}X_{45}|X_{39}X_2$
$X_{24} \rightarrow X_3X_{31}|X_{18}X_{32}|X_{37}X_{17}|X_{14}X_{15}|X_{11}X_1|X_{23}X_{29}|X_{46}X_{41}$
$X_{27} \rightarrow' here'$
$X_{26} \rightarrow X_{30}X_{14}|X_{21}X_{12}|X_{10}X_{28}|X_{36}X_{47}|X_{39}X_{35}|X_{25}X_{38}|X_{40}X_{42}$
$X_{21} \rightarrow' He'|X_{30}X_3$
$X_{20} \rightarrow' will'$
$X_{23} \rightarrow X_3X_{45}|X_{11}X_2$
$X_{22} \rightarrow X_6X_{27}$
$X_8 \rightarrow X_{21}X_{45}|X_{39}X_2$
$X_9 \rightarrow X_{20}X_5|X_{45}X_4|X_{34}X_{42}$
$X_2 \rightarrow' be'$
$X_3 \rightarrow Mr'$
$X_1 \rightarrow X_2X_{29}|X_{19}X_{16}|X_5X_{15}|X_{19}X_{41}|X_{35}X_{15}$
$X_6 \rightarrow' was'|'went'$
$X_4 \rightarrow X_{13}X_{42}$
$X_5 \rightarrow X_2X_4|X_{19}X_{42}$
$X_{32} \rightarrow X_{27}X_{17}|X_{28}X_{15}$
$X_{33} \rightarrow X_{21}X_9|X_{39}X_5|X_8X_4|X_{40}X_{42}$
$X_{30} \rightarrow Smith$
$X_{31} \rightarrow X_6X_{32}|X_{22}X_{17}|X_{12}X_{15}|X_{20}X_1|X_{45}X_{29}|X_{43}X_{41}$
$X_{36} \rightarrow X_{30}X_{37}|X_{21}X_{22}|X_{10}X_{27}$
$X_{37} \rightarrow X_3X_{22}|X_{18}X_{27}$
$X_{34} \rightarrow X_{20}X_{19}|X_{45}X_{13}$
$X_{35} \rightarrow X_2X_{38}|X_{19}X_{42}$
$X_{38} \rightarrow X_{13}X_{42}$
$X_{39} \rightarrow X_{30}X_{11}|X_{21}X_{20}$
$X_{18} \rightarrow X_3X_6$
$X_{19} \rightarrow X_2X_{13}$
$X_{10} \rightarrow X_{30}X_{18}|X_{21}X_6$
$X_{11} \rightarrow X_3X_{20}$
$X_{12} \rightarrow X_6X_{28}|X_{22}X_{47}|X_{20}X_{35}|X_{45}X_{38}|X_{43}X_{42}$
$X_{13} \rightarrow' there'|'gone'$
$X_{14} \rightarrow X_3X_{12}|X_{18}X_{28}|X_{37}X_{47}|X_{11}X_{35}|X_{23}X_{38}|X_{46}X_{42}$
$X_{15} \rightarrow' morning'|'evening'$
$X_{16} \rightarrow X_{42}X_{15}$
$X_{17} \rightarrow X_{47}X_{15}$

Étant donné le petit échantillon d'apprentissage, il est facile de voir que le résultat obtenu est une grammaire très redondante du langage cible visé et qu'il existe une représentation plus simple pour visualiser la généralisation de l'échantillon d'apprentissage en exhibant les classes de substituabilité. Avec notre algorithme, nous obtenons la grammaire directement réduite suivante, illustrant le fort gain de compression et de lisibilité qui peut être obtenu par notre approche :

$$
\begin{aligned}
S &\rightarrow X_3 X_4 X_2 \\
X_1 &\rightarrow was \mid went \\
X_2 &\rightarrow morning \mid evening \\
X_3 &\rightarrow He \mid Mr\ Smith \\
X_4 &\rightarrow will\ be\ X_5\ tomorrow \mid X_1\ here\ yesterday \\
X_5 &\rightarrow there \mid gone
\end{aligned}
$$

3.4 Expérimentations

Dans cette section, nous présentons dans un premier temps une comparaison des temps d'exécution de l'algorithme ReGLiS et du premier algorithme naïf pour montrer l'apport de notre approche en terme de complexité. Puis, nous présenterons le protocole complet d'apprentissage à partir d'un ensemble de séquences de protéines, qui doit inclure un pre-processing automatique des séquences afin d'intégrer les spécificités des données. Les expérimentations sur un jeu de données réelles terminent cette présentation.

3.4.1 Comparaison des temps d'exécution sur des données simulées

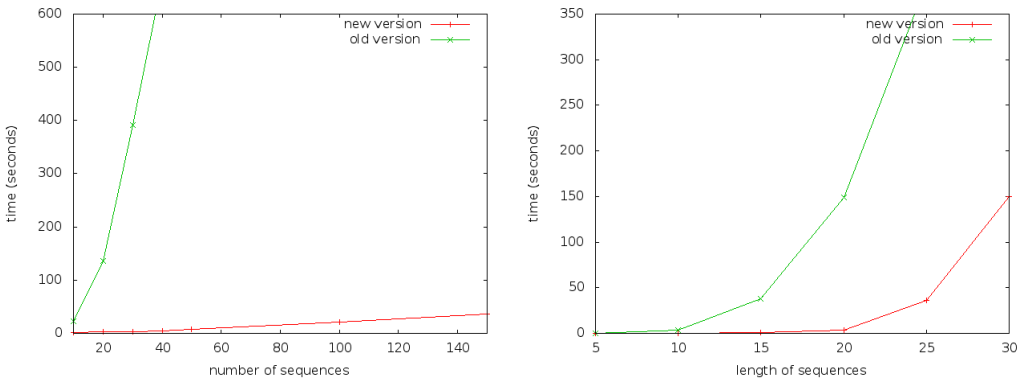
Nous avons lancé des tests sur des ensembles de données simulées pour comparer l'algorithme ReGLiS avec la version précédente. Le gain de temps d'exécution a été estimé sur ces échantillons d'apprentissage en augmentant le nombre de séquences et leur taille, de façon à se rapprocher des conditions rencontrées dans la tâche de classification de protéines qui nous intéressent.

Pour l'expérience, une séquence aléatoire de taille 20 sur un alphabet de 40 symboles est générée de façon aléatoire et triée selon un ordre arbitraire sur l'alphabet comme pour le résultat d'un recodage de séquence d'une protéine par rapport à un alignement multiple local partiel.

Pour obtenir le nombre de séquences similaires voulues, de nouvelles séquences de symboles ont été générées de manière itérative en remplaçant chaque symbole de la dernière chaîne générée avec une probabilité de 25 % en un symbole aléatoire de l'alphabet et en triant ensuite la séquence selon un ordre défini sur l'alphabet. Pour étudier l' importance de la longueur, la même procédure de génération a été utilisée, mais avec le nombre de séquences fixées à 20 et la taille de l' alphabet fixée à deux fois la longueur des séquences de l'échantillon, cette dernière valeur ayant été choisie à partir de l'observation pratique d'expériences sur les séquences de protéines.

Les résultats sont présentés dans la figure 3.7 qui montre qu'il y a gain de temps très net quand le nombre de séquences croît. Le paramètre de longueur a un impact fort sur le temps d'exécution comme cela était attendu dans l'analyse de complexité. La courbe montre que, pour une quantité donnée de ressources, le nouvel algorithme tourne 10 fois

plus vite que l'ancien pour des séquences de longueurs égales, une différence qui peut être d'une grande importance dans la pratique.

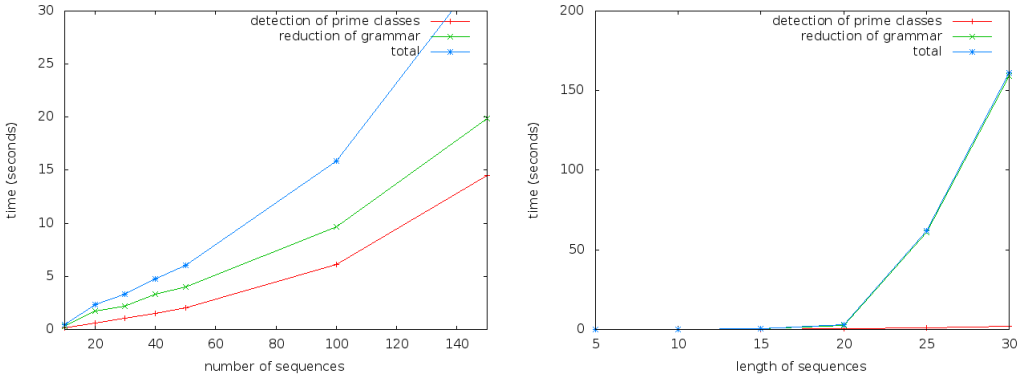


(a) en fonction du nombre de séquences dans l'échantillon d'apprentissage

(b) en fonction de la taille des séquences dans l'échantillon d'apprentissage

Fig. 3.7 – Comparaison des temps d'exécution entre l'ancienne et la nouvelle version de l'algorithme

Dans la figure 3.8, les temps pour les deux parties principales de l'algorithme, la détection des classes et la réduction des classes, ont été extraits. Comme attendu, la complexité est déterminée par la partie effectuant la réduction et croît fortement en fonction de la longueur des séquences d'entrée. La détection des classes premières semble avoir une complexité linéaire, un comportement empirique meilleur que celui du pire cas théorique en complexité.



(a) en fonction du nombre de séquences dans l'échantillon d'apprentissage

(b) en fonction de la taille des séquences dans l'échantillon d'apprentissage

Fig. 3.8 – Temps pour la détection des classes K-premières, réduction de la grammaire et temps total

Après avoir montré que notre algorithme permet de traiter un plus grand nombre

de séquences en un temps raisonnable, nous avons procédé à des expériences sur des ensembles de séquences de protéines réelles.

3.4.2 Processus d'apprentissage sur les séquences de protéines

Afin de traiter les données particulières que sont les séquences biologiques, nous avons mis en place une chaîne de traitement spécifique.

La figure 3.9 fournit une vue d'ensemble des étapes de pré et post-processing automatiques nécessaires pour appliquer l'étape de généralisation principale de ReGLiS sur des séquences de protéines.

Notre approche repose sur la détection de similarités locales entre les séquences par la construction d'un alignement multiple local partiel (PLMA) des séquences. Chaque région fortement conservée dans un PLMA (aussi appelé bloc) sera l'un des symboles de recodage des séquences. [3]

.

En plus des blocs, il est nécessaire de prendre en compte l'existence de sous-séquences qui n'apparaissent dans aucun bloc et c'est ce qui explique la procédure assez compliquée pour obtenir une grammaire pratique directement capable d'analyser des séquences de protéines à partir de la grammaire apprise.

Le paragraphe suivant détaille les étapes de la procédure.

Les séquences sont recodées selon les blocs qui intersectent avec la séquence (step $b.1$) et en parallèle l'information de la composition en acides aminés pour chaque bloc est retenue dans une grammaire G_a (step $b.2$). Pour chaque bloc B de longueur l, la règle suivante est créée :

$$B \rightarrow P_1 ... P_l,$$

et pour chaque acide aminé A à la position p dans le bloc, on ajoute une règle :

$$P_p \rightarrow N_A.$$

En réalité, c'est un peu plus complexe car une connaissance a priori sur les protéines peut être introduite dans cette grammaire. Plus précisément, nous avons ajouté un modèle d'erreur simple, pour autoriser des séquences proches, sur la base de la matrice de substitution d'acides aminés de type Blosum62, qui marque la possibilité qu'un acide aminé donné puisse être remplacé par un autre sans perte de fonctionnalité de la protéine. Ainsi, pour chaque paire d'acides aminés (A,C) pour lesquels le score reflète qu'une mutation est plus fréquente que le hasard, la règle suivante est ajoutée :

$$N_A \rightarrow C.$$

[3] Dans la pratique, nous avons utilisé la ligne de commande suivante pour tous les PLMA : Paloma -i foo.fasta -o foo.plma –block-mode maxWeightFragments -transClosure -t 5 -M 7 -q 2

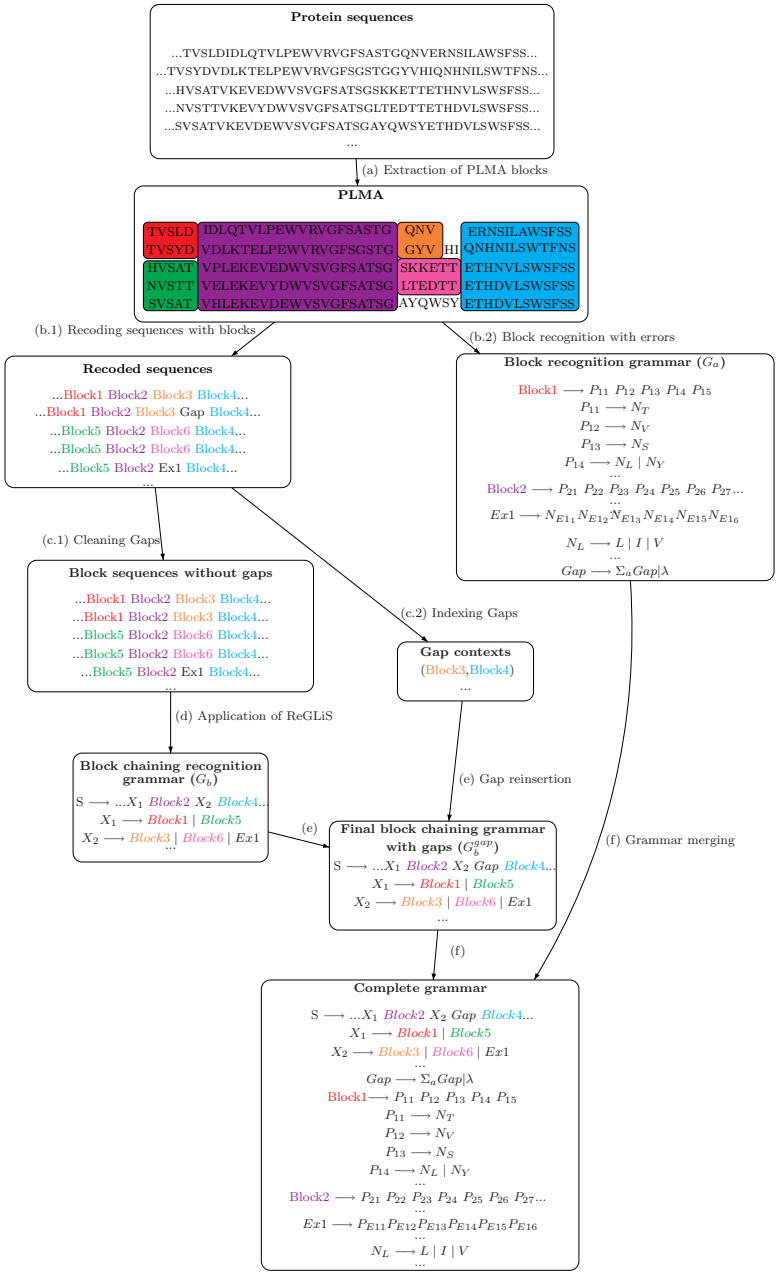


Fig. 3.9 – Vue d'ensemble de l'apprentissage de grammaires hors-contexte à partir d'un échantillon de séquences protéiques

C'est une façon d'agrandir artificiellement la taille de l'échantillon d'apprentissage qui est généralement trop petit pour fournir cette information de mutations sur des séquences de

protéines.

Le raisonnement derrière le recodage en blocs est que les parties non conservées sont susceptibles d'être inintéressantes pour la caractérisation d'une famille de protéines. Ainsi, comme dans [Ker08], il faut fixer le traitement des facteurs n'appartenant pas à un bloc. Si un facteur se trouve entre deux blocs et qu'aucun bloc n'est trouvé entre ces deux blocs dans d'autres séquences, il est considéré comme un gap et une règle est dédiée à sa reconnaissance :

$$Gap \rightarrow \Sigma_a Gap | \lambda.$$

D'autres chaînes qui permettent un court circuit d'un ou de plusieurs blocs sont appelées « exceptions ». Afin d'éviter une trop grande généralisation, nous avons gardé l'exception $E_0 \ldots E_n$ comme des nouveaux blocs et ajouté une règle dédiée dans la grammaire :

$$E \rightarrow N_{E_0} \ldots N_{E_n}.$$

Avant d'appliquer ReGLiS sur les séquences recodées, les gaps sont éliminés (étape $c.1$) pour éviter une généralisation dépourvue de sens (un contexte avec seulement des gaps n'est pas significatif). Le contexte immédiat (blocs gauche et droit) de chaque gap est stocké (étape $c.2$) pour le réinsérer ultérieurement.

L'étape principale d applique l'algorithme sur les séquences de blocs de PLMA sans gaps et produit la grammaire G_b.

Les gaps sont réinsérés dans la grammaire durant la phase e en remplaçant chaque règle $L \rightarrow \ldots \alpha_i \alpha_{i+1} \ldots$ dans G_b quand le dernier symbole de α_i et le premier de α_{i+1} forment le contexte d'un gap, par la règle :

$$L \rightarrow \ldots \alpha_i \ Gap \ \alpha_{i+1} \ldots$$

Ceci conduit à une grammaire G_b^{gap} qui est fusionnée avec G_a en changeant les blocs terminaux par des non-terminaux (étape f).

3.4.3 Résultats d'apprentissage sur des familles de protéines

Pour tester notre approche, nous avons décidé de considérer l'ensemble des jeux de données utilisés dans [DN09]. Cet article utilise des grammaires algébriques stochastiques pour modéliser des familles de séquences de protéines en introduisant des connaissances sur leur structure spatiale. Les résultats sont présentés dans la table 3.4.3.

Lorsque cela est possible, nous avons également comparé nos résultats avec des résultats obtenus en utilisant des expressions régulières de Prosite. Il est souhaitable d'obtenir une spécificité élevée car les expériences biologiques sur les protéines sont coûteuses et un nombre limité de candidats peuvent être évalués et validés dans la pratique.

Les modèles Prosite couvrent généralement environ 10 positions alors que la protéine entière fait en général 300 caractères de long. En conséquence, un tel modèle a un bon rappel mais une faible précision en raison de sa généralité. Au contraire, notre méthode prend en compte les séquences entières. La grammaire correspondante a donc une grande

spécificité. Le point intéressant est que, malgré leur spécificité, le niveau de rappel obtenu par ces grammaires semble demeurer relativement élevé.

Dans toutes les expériences, nous utilisons une validation croisée avec 10 échantillons et choisissons des paramètres k et l relativement faibles puisque l'apprentissage se fait sur des séquences de blocs qui sont plus courtes que les séquences d'acides aminés d'origine.

Une hypothèse raisonnable est qu'une connaissance a priori peut exister sur la longueur de contextes pertinents dans l'application cible. Dans le cas des protéines, de petits contextes sont attendus pour les interactions qui concernent les acides aminés. En pratique ces valeurs vont de 3 à 7. Dans tous les cas, il faut éviter des valeurs plus élevées, car la substituabilité globale est moins présente, ce qui explique les mauvais résultats obtenus avec une substituabilité non locale.

Dans l'ensemble, on peut observer que la substituabilité locale, en utilisant des paramètres k et l de valeurs bien choisies, améliore considérablement le rappel sans perdre en précision contrairement à la substituabilité globale. Les grammaires stochastiques obtiennent de meilleurs résultats en termes de F-mesure, sauf si la précision est fixée à un niveau élevé. Dans un tel cas, le rappel obtenu par ReGLiS est généralement préférable. En outre, nos meilleurs résultats de rappel sont comparables à Prosite, qui est considéré comme un outil de prédiction expert.

Tous les détails des expériences et des résultats sont disponibles sur http://www.irisa.fr/dyliss/reglis.

Conclusion

Dans ce chapitre, nous avons proposé l'introduction des langages localement et contextuellement substituables ainsi qu'un algorithme qui permet d'apprendre en utilisant les principes de généralisation de ces langages.

Nous avons également conçu un algorithme générique capable de conserver la structuration du langage appris qui apprend directement une grammaire réduite, une forme canonique d'un langage substituable, identifiable à la limite polynomialement en temps et données fixés.

Notre dernier apport a été la mise au point d'un pipeline complet permettant d'automatiser la prise en compte des spécificités de nos données de séquences de protéines pour les intégrer à la grammaire apprise.

Nous avons ainsi réussi à caractériser une famille d'enzymes grâce à une grammaire algébrique.

Un dernier problème reste cependant à traiter. Qu'arrive-t-il lorsque nous disposons d'une superfamille entière (avec ou sans annotation concernant les familles) ? Caractériser un ensemble de familles directement au niveau d'une superfamille fait ressurgir de nombreux problèmes et provoque une surgénéralisation du langage voulu. Le prochain chapitre explore l'utilisation des concepts formels afin de classifier un ensemble de séquences d'une même superfamille pour pouvoir ensuite appliquer l'inférence grammaticale au niveau adéquat des familles de protéines.

	Zinc finger			MPI phos.		
	Précision	Rappel	F-meure	Précision	Rappel	F-mesure
substituable	1	0.1	0.36	1	0.15	0.26
(3,3)-LS	1	0.2	0.33	1	0.5	0.67
(4,4)-LS	1	0.25	0.4	1	0.6	0.75
(5,5)-LS	1	0.33	0.5	**1**	**0.67**	**0.8**
(6,6)-LS	1	0.5	0.67	1	0.62	0.77
(7,7)-LS	**1**	**0.55**	**0.7**	1	0.53	0.69
CFG stochastiques [DN09]	1 0.15 0.75	0.1 1 0.87	0.18 0.26 0.85	1 0.5 0.98	0.3 1 0.89	0.46 0.67 0.93
	PS00219			PS00063		
	Précision	Rappel	F-mesure	Précision	Rappel	F-mesure
Substituable	1	0.2	0.33	1	0.23	0.37
(3,3)-LS	**1**	**0.72**	**0.84**	1	0.58	0.73
(4,4)-LS	1	0.7	0.82	1	0.6	0.75
(5,5)-LS	1	0.68	0.8	1	0.66	0.8
(6,6)-LS	1	0.6	0.75	**1**	**0.7**	**0.82**
(7,7)-LS	1	0.5	0.67	1	0.65	0.79
Prosite	1	0.6	0.75	1	0.8	0.89
CFG stochastiques [DN09]	- - 1	- - 1	- - 1	1 0.1 0.79	0.05 1 0.65	0.1 0.18 0.71

Tab. 3.2 – Prédiction des classes des séquences obtenue grâce aux grammaires apprises (substituables, localement substituables, Prosite et CFG stochastiques). Les meilleurs résultats apparaissent en gras.

Chapitre 4

Classification de séquences par analyse de concepts formels

L'inférence grammaticale d'une famille d'enzymes suppose qu'on dispose d'un échantillon correctement étiqueté de séquences pour chacune des familles. En pratique, s'il est assez facile d'annoter chaque séquence par la superfamille à laquelle elle appartient, l'identification des familles reste un problème difficile. Dans ce chapitre, nous nous intéressons à la prédiction des familles fonctionnelles au sein d'une superfamille d'enzymes.

Les fonctions des enzymes peuvent être associées à des positions particulières dans leurs séquences, correspondant à des acides aminés spatialement proches impliqués dans les interactions moléculaires et qui contraignent la structure spatiale de ces enzymes. Celles-ci participent à la liaison spécifique à un substrat, ou sont impliquées dans le mécanisme catalytique. Dans la pratique, des facteurs courts extraits de séquences d'enzymes partageant une même activité connue peuvent aider à identifier ces sites actifs.

Le but du travail que nous présentons ici est d'utiliser des ensembles de séquences étiquetées (numéro EC) et non étiquetées d'une superfamille afin d'inférer les familles auxquelles elles appartiennent. Les séquences non étiquetées sont des séquences à classer. Les séquences étiquetées servent d'exemples positifs pour chaque classe d'enzyme présente dans l'échantillon mais peuvent également être vues comme des exemples négatifs pour les autres classes, aidant à la découverte de signatures spécifiques à chaque classe.

Nous présentons une méthode de prédiction qui peut être utilisée pour trois buts principaux :

- classer les séquences non étiquetées grâce à l'alignement et à un certain nombre de blocs partagés avec des séquences étiquetées (classification supervisée) ;
- découvrir de nouveaux groupes de séquences dans lesquels aucune n'est étiquetée et qui vont former une nouvelle famille (classification non supervisée) ;
- découvrir les blocs discriminants de chacune des familles présentes dans l'échantillon (y compris les nouvelles découvertes) afin d'inférer une signature pour chaque famille (inférence d'un modèle de signature discriminant)

4.1 Analyse de concepts formels à partir d'un PLMA

Nous utilisons ici l'analyse de concepts formels en recodant les séquences sous forme d'attributs associés à la présence/absence de certains motifs détectés comme essentiels au sein de la superfamille et des familles qui la composent grâce à un alignement local multiple partiel.

4.1.1 Codage des séquences d'enzymes

Afin de détecter les similarités partagées par les différentes séquences de l'échantillon, nous utilisons un alignement partiel local multiple des séquences (cf section 1.2).

Le calcul d'un PLMA est la première étape réalisée par Protomata-Learner ([Ker08]). Alors que dans Protomata-Learner, il est important de régler les paramètres d'alignement pour obtenir le niveau de généralisation souhaité, dans cette étude nous avons seulement utilisé les paramètres par défaut pour représenter chaque séquence par une séquence de blocs issus de l'alignement.

Chaque bloc b_i détecté dans un PLMA P est alors vu comme un attribut dont la présence/absence peut caractériser la séquence. Formellement, chaque séquence s est recodée en un vecteur : $s = [b_1, b_2 ... b_i ... b_n]$, tel que

$$b_i = \begin{cases} 1 & \text{si le bloc i est présent dans la séquence } s \text{ dans P} \\ 0 & \text{sinon.} \end{cases}$$

A ce stade, il est important de noter que les nouvelles séquences à annoter, les séquences non étiquetées, doivent également être alignées et recodées avec les séquences dont la classe est connue.

Le but est alors de définir les concepts ⟨séquences,blocs⟩ et d'en déduire les blocs caractérisant une famille fonctionnelle.

4.1.2 Observation d'un lien séquence/structure sur une superfamille multi-fonctionnelle

Dans un premier temps nous avons expérimenté ce procédé sur un exemple introductif (la superfamille des GH16) en introduisant la connaissance *a priori* de l'appartenance d'une séquence à une famille afin de déterminer si certains concepts liés à la fonction de l'enzyme apparaissent dans un alignement.

Les GH16 [CCR+09, HCB+10] forment une superfamille très étudiée et bien connue au niveau structural. Cela nous a permis de valider l'hypothèse d'un lien existant entre les concepts repérés et l'activité des enzymes.

Cette superfamille possède notamment un grand nombre de substrats différents et donc différentes activités enzymatiques pour une même structure et une même signature du site actif sur les séquences : $E-x(1)-D-x(1,2)-E$.

Parmi les différents substrats de GH16, on trouve en particulier (l'activité enzymatique liée à ce substrat est indiquée entre parenthèses.) :

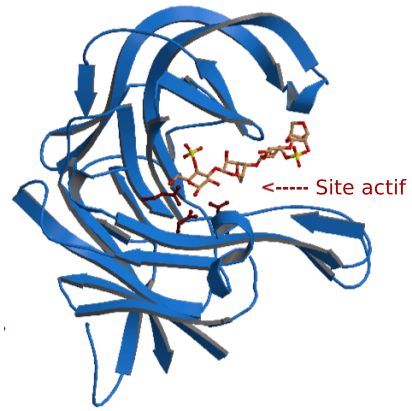


Fig. 4.1 – Structure d'une GH16 en complexe avec son substrat

- l'agar (EC. 3.2.1.81)
- le kappa carraghenan (EC. 3.2.1.83)
- le porphyran (EC. 3.2.1.178)
- le keratan sulphate (EC. 3.2.1.103)
- le xyloglucan (EC. 3.2.1.151)
- le laminarin (EC. 3.2.1.39)
- le lichenan (EC. 3.2.1.73)
- ...

Un extrait d'alignement est fourni en annexe A.

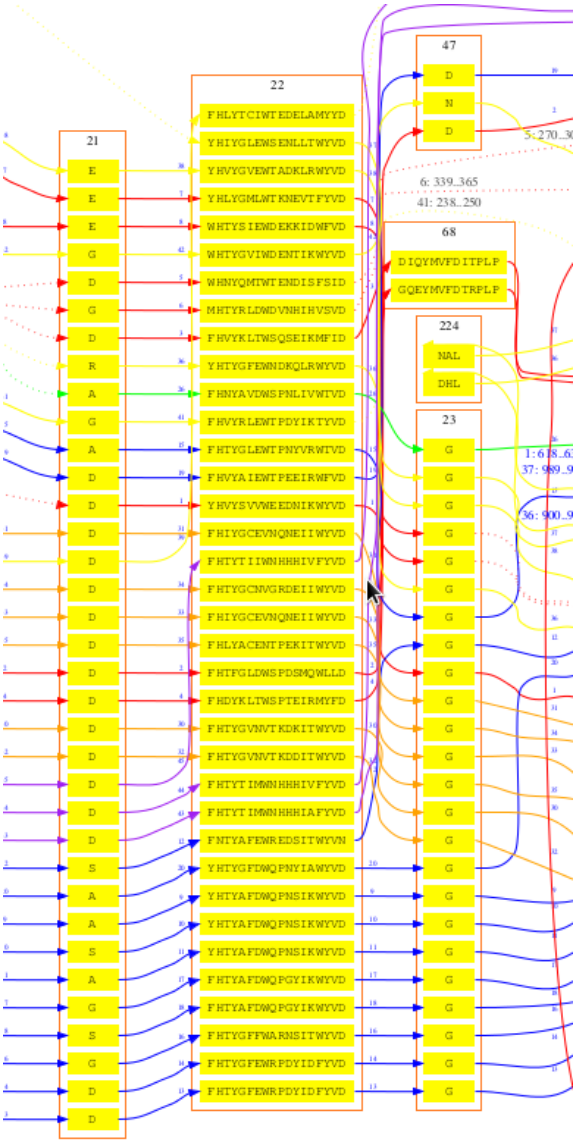
Sur celui-ci, chaque séquence est représentée par une suite d'arcs parcourant les différents blocs présents dans cette séquence. Si la famille de la séquence est connue, la séquence d'arcs correspondante sur l'alignement apparaît colorée, une couleur représentant une famille.

On peut déduire de cet alignement un contexte formel (S,B,I) où S (les objets) est l'ensemble des séquences, B (les attributs) l'ensemble des blocs et $(S_i,B_j) \in I$ si $S_i[j] = 1$ dans le recodage de la séquence S_i en terme de présence/absence de blocs.

On construit les concepts formels issus de ce contexte pour le classement des séquences en familles.

Deux types de blocs apparaissent dans ces alignements :

- des blocs conservés par toutes les séquences découlant de l'homologie des séquences au niveau de la superfamille qui participent à la structure générale de l'enzyme mais n'intervenant pas dans la réalisation de sa fonction. Un exemple est donné sur la figure 4.2(a). Les attributs du concept associé à ce bloc sont projetés sur la structure 3D en figure 4.2(b).
- des blocs clés apparemment liés directement à la fonction de l'enzyme comme sur la figure 4.3(a). Les attributs du concept associé à ce bloc sont projetés en figure 4.3(b).



(a) bloc commun à plusieurs familles

(b) projection du concept associé sur la structure 3D

Fig. 4.2 – Motif conservé sur la séquence et projection sur la structure 3D. 4.2(a) : chaque couleur de séquence représente une famille, ici toutes les séquences passant par le bloc central n'appartiennent pas à la même famille. 4.2(b) : la partie de séquence correspondante au bloc 22 de la figure 4.2(a) a été colorée en orange, on peut remarquer qu'elle se trouve loin du site actif et ne semble donc pas impliquée dans la fonction de l'enzyme.

Sur cet exemple, on remarque que les concepts formels issus de cet alignement qui semblent liés à l'activité (dont les séquences appartiennent à une même famille) sont généralement situés à proximité du site actif et possèdent différents blocs éloignés sur la séquence mais proches sur la structure alors que les autres blocs sont plutôt éloignés du site actif, isolés et semblent être liés à la structure même de l'enzyme.

On dispose ici de plusieurs types d'informations pour décider des concepts représentant une famille : premièrement l'étiquetage des séquences qui permet d'associer certains blocs à certaines fonctions et deuxièmement une information de structure qui permet de vérifier que les blocs appartenant à un concept interagissent réellement spatialement.

Dans la prochaine section nous utiliserons essentiellement l'information des séquences étiquetées, qui est l'information la plus couramment disponible afin de choisir les concepts intéressants, c'est-à-dire discriminant au niveau fonctionnel.

4.2 Annotation via l'analyse de concepts formels

Étant donnée une superfamille connue, nous considérons la question de la classification d'un ensemble de nouvelles séquences d'enzymes (l'ensemble non étiqueté) au niveau de la famille par rapport à un ensemble de séquences qui ont déjà été classées (l'ensemble étiqueté). Ce travail a fait l'objet d'un article dans la conférence ICFCA'14 [CGG+14].

4.2.1 Formalisation du problème de classification

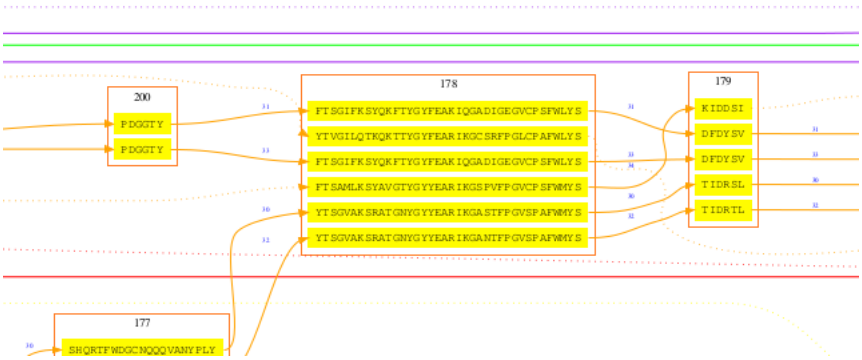
Une fois que toutes les séquences de protéines ont été représentées par des vecteurs de présence de blocs, il reste à attribuer une classe (une fonction) à chaque séquence non étiquetée. On note que le cadre défini ici autorise qu'une séquence possède plusieurs fonctions et appartienne donc à plusieurs classes.

Il s'agit soit de classes connues, soit de nouvelles classes qui n'ont pas encore été observées. En effet, la présence simultanée de blocs spécifiques dans l'ensemble non étiqueté peut être la trace de l'existence d'une nouvelle classe.

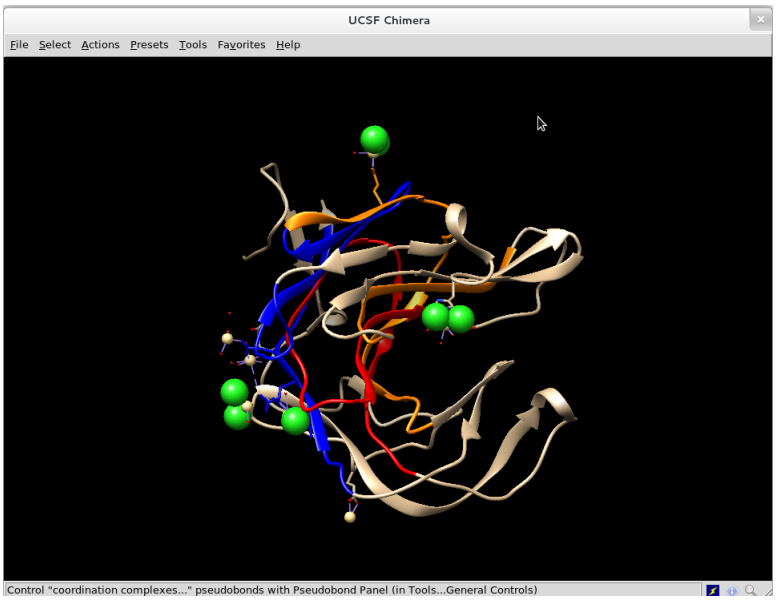
Plus formellement, le problème d'apprentissage consiste alors à apprendre une fonction de classification f à partir d'un ensemble d'apprentissage qui contient des séquences exemples $S = \{s_1, s_2, ..., s_i, ..., s_n\}$ décrites par un ensemble fini d'attributs (ici des blocs) $A = \{a_1, a_2, ..., a_j, ..., a_m\}$ et appartenant à une ou plusieurs classes représentant les différentes familles enzymatiques $C = \{c_1, c_2, ..., c_k, ..., c_p\}$. L'ensemble d'apprentissage caractérise alors une certaine fonction f à déterminer.

L'ensemble d'apprentissage, de taille n, est une suite de couple (s_i, C_k) où $1 \leq i \leq n$, $s_i \in S$, $C_k \subseteq C$, $|C_k| \geq 0$. Dans le cas où $|C_k| \geq 1$ (séquences étiquetées), la fonction f est telle que $C_k = f(s_i)$. Chaque exemple s_i est représenté par un vecteur $(s_{i_1}, ..., s_{i_m})$ où s_{i_j} est la valeur de s_i pour l'attribut $a_j \in A$, qui indique la présence d'un bloc j sur la séquence i. C_k est l'ensemble des classes pouvant être attribuées à s_i.

Le problème est de classer les séquences non étiquetées en étendant cette fonction f. On cherche alors une fonction $g \sim f$ telle que $g(s_x) = C_x$. $|C_x| \geq 1$ pour n'importe quelle séquence $s_x \in S$ représentée par la présence ou l'absence des blocs de A.



(a) blocs caractéristique de la famille des kappa carraghenases (chaque couleur de séquence représente une famille, ici toutes les séquences passant par le bloc 178 appartiennent donc à la même famille)



(b) projection du concept maximal associé sur la structure 3D : la partie de séquence correspondante au bloc B de la figure 4.3(a) a été colorée en orange, les fragments colorés en rouge sont ceux correspondant aux autres blocs appartenant au concept $C = \langle B', B'' \rangle$. En bleu sont représentés les blocs non présents dans C appartenant aux concepts supérieurs à C (incluant strictement les séquences de C)

Fig. 4.3 – Bloc fonctionnel B caractéristique et projection du concept $\langle B', B'' \rangle$ associés sur la séquence 3D

L'apprentissage de cette fonction est soit supervisé si $C_x \subseteq C$ (la famille de la séquence à classer se trouve dans l'échantillon d'apprentissage), soit non supervisé dans le cas contraire (détection d'une nouvelle famille).

Une approche naturelle pour une telle tâche est de construire une classification de toutes les séquences en se servant des attributs binaires (présence de blocs) et de décider la classe des séquences non étiquetées selon leur place parmi les autres dans l'arbre de classification. Cela nécessite de définir une mesure de similarité sur l'ensemble des attributs binaires, et de fixer un seuil pour discriminer les groupes significatifs. Des problèmes se posent rapidement en essayant de suivre cette approche : le nombre d'attributs peut grandement varier d'une superfamille à l'autre et d'une séquence à l'autre au sein d'une même famille. Une décision prise sur des arguments statistiques n'est pas pleinement satisfaisante car il est difficile de fixer des valeurs universelles pour les paramètres nécessaires et, finalement, un biologiste doit vérifier les annotations sur la base de la logique de l'argumentation, sa propre connaissance, et en outre la caractérisation biochimique des séquences d'intérêt.

Une approche par analyse de concepts formels permet de s'adapter plus facilement à chaque ensemble d'apprentissage (chaque superfamille) car elle couvre l'ensemble des classifications possibles en extrayant les concepts discriminants sur le produit $\textit{séquences} \times \textit{blocs}$.

L'application de l'analyse de concepts formels à la classification supervisée a déjà été proposée dans la littérature dans différents domaines. La grande majorité des travaux produisent un classifieur à partir du treillis de concepts construits sur un ensemble d'objets étiquetés. Celui-ci est utilisé dans une deuxième étape pour attribuer une classe à de nouveaux objets de classe inconnue. Le treillis de concepts est idéal pour la recherche de règles de décision. Cette recherche peut être faite en transformant le treillis initial ($closed\ label\ lattice$, [WLQ11]). La façon la plus efficace de générer un classifieur est de générer les règles ou les nœuds de décision directement à partir de concepts sélectionnés dans le treillis. Par exemple, il est possible de construire un arbre de décision directement à partir du treillis [Kov07] ou via le calcul des intersections de concept dans le treillis [Sah95].

Dans toutes ces études, l'ensemble des objets non étiquetés est utilisé seulement une fois que le classifieur est construit. En revanche, dans [IY13], le treillis est construit simultanément sur l'ensemble des objets (étiquetés ou non) afin de centrer la recherche sur les liens entre les objets connus et inconnus. Pour chaque concept, un score est alors calculé pour estimer la plausibilité qu'un concept représente un ensemble de voisins (objets appartenant à une même classe). L'étiquette de la classe d'objets est prise en compte dans le score. La méthode sélectionne d'abord le concept le plus discriminant pour chaque objet non étiqueté et le classe par rapport à l'étiquette des objets présents dans ce concept.

Dans notre étude, les attributs sont représentés par les blocs utilisés pour le recodage des séquences. Il y a deux types d'objets : les séquences d'enzymes étiquetées et celles non étiquetées. L'idée est alors d'introduire les connaissances sur les classes des objets directement dans le contexte formel. Ceci peut être résolu simplement en ajoutant les valeurs de classe comme de nouveaux objets. Chaque fois qu'un bloc b est observé dans une séquence de classe c, la paire (b,c) est ajoutée au contexte formel. Introduire les

classes dans le contexte comme des objets permet d'avoir la bonne sémantique pour la relation binaire : cela reflète la présence de chaque bloc dans une séquence d'enzyme et une classe. En pratique, il est seulement nécessaire de produire des concepts possédant au moins une séquence non étiquetée. La taille de la relation obtenue pour ce contexte reste suffisamment petite pour produire l'ensemble du treillis de concepts formels. La procédure d'affectation de classes est alors basée sur l'exploration du treillis.

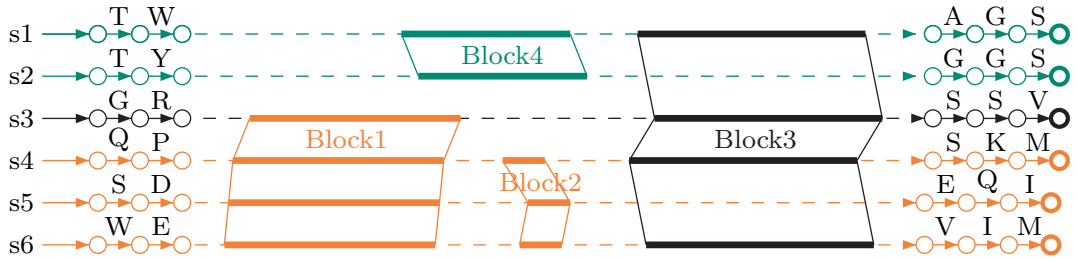
Pour illustrer ce procédé, les figures 4.4(a), 4.4(b) et 4.4(c) représentent des alignements locaux partiels multiples et dans chaque figure, les séquences de couleur (par exemple, $s1$ et $s2$) sont des séquences étiquetées tandis que les séquences en noir (par exemple $s3$) sont non étiquetées et doivent être classées. Sur la figure 4.4(a) la séquence $s3$ possède une classe spécifique correspondant à la classe orange et peut donc être classée sans ambiguïté grâce à un bloc spécifique. Le contexte formel issu de cet alignement est montré en figure 4.1. Cependant, sur la figure 4.4(b) il y a deux concepts spécifiques (orange et vert), et l'affectation de classe de séquence s_3 est ambiguë. La figure 4.4(c), fournit un exemple d'une séquence $s3$ restée non classée parce que l'alignement multiple n'a pas trouvé de bloc commun avec une autre séquence. Sur la même figure, une nouvelle famille est formée avec un bloc caractéristique impliquant uniquement des séquences non étiquetées : $\{S4, S5, S6\} \times \{Bloc1, Bloc2, Block3\}$. Nous détaillons maintenant la recherche de ces nouvelles classes.

Sequence x Block	Block1	Block2	Block3	Block4
S_1				X
S_2				X
$Family_2$				X
S_3	X		X	
S_4	X	X	X	
S_5	X	X		
S_6	X	X	X	
$Family_1$	X	X	X	

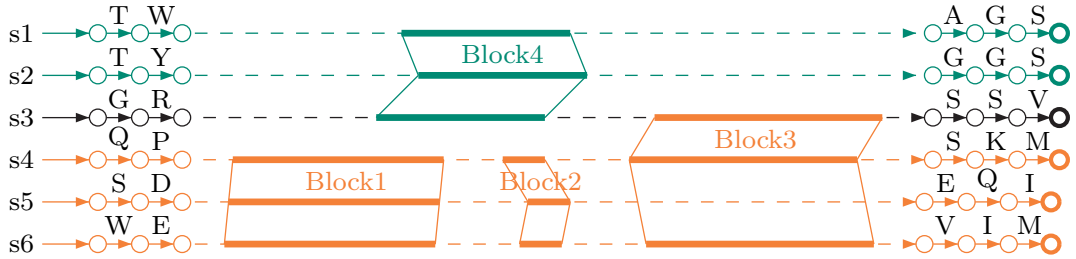
Tab. 4.1 – Contexte formel correspondant à l'alignement de la figure 4.4(a)

Ici, on définit $L \subseteq S$ comme l'ensemble des objets étiquetés et $U \subseteq S$ celui des objets non étiquetés. Soit I la relation binaire sur $(L+U+C) \times A$ et $\mathscr{B}((L+U+C), A, I)$ le treillis de concept, le problème est de trouver un ensemble minimal de nouvelles classes N et un argument permettant d'assigner les classes de $N \cup C$ aux éléments de U sur la base de $\mathscr{B}((L+U+C), A, I)$.

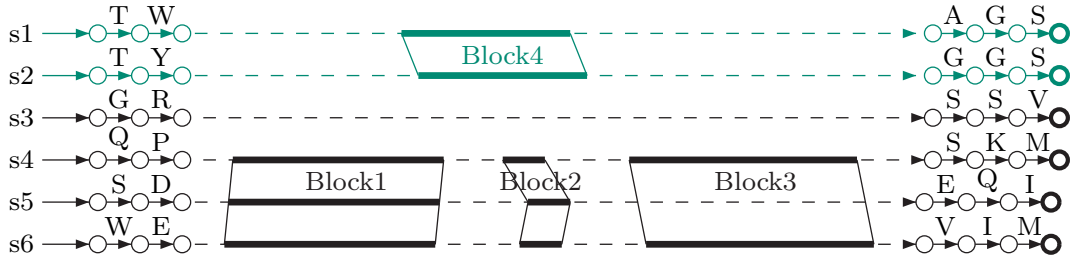
Dans un premier temps, nous cherchons à repérer les blocs spécifiques de certaines classes. Pour cela, nous divisons L en sous-ensembles $L_i, i = 1, ..., m$ de même classe i et une partition de A en construisant $m+1$ sous-ensembles, éventuellement vides, $A_i, i = 0, m$ correspondant à des attributs présents uniquement dans les éléments de L_i, avec $A_0 = A \setminus \cup_{i=1}^{n} A_i$. A_0 correspond à l'ensemble des blocs impliqués dans plusieurs classes ou au



(a) $s3$ est classée



(b) $s3$ est ambiguë



(c) $s3$ est n classée et $\{s4, s5, s6\}$ forment une nouvelle famille

Fig. 4.4 – Exemples d'alignements locaux partiels multiples avec des séquences étiquetées (colorées) et non étiquetées (noires)

contraire dans aucune des classes connues. Nous noterons $A_s = \cup_{i=1}^{n} A_i$ l'ensemble des blocs spécifiques détectés.

Notre méthode vise à maximiser la spécificité des décisions de classification et propose donc plusieurs niveaux de qualité de classement à cette fin (cf figure 4.5) : les séquences classées grâce à des blocs spécifiques appartenant à $A_i, i = 1,..,m$, les séquences classées sans la présence d'un bloc spécifique, les séquences ambiguës passant par des blocs spécifiques de plusieurs familles, les séquences ambiguës ne passant pas par des blocs spécifiques à une famille. Enfin, nous terminons par les séquences qu'il n'a pas été possible de classer, ne possédant aucune classe compatible (soit parce que ses blocs appartiennent à A_0, soit parce qu'elles ne possèdent aucun bloc).

4.2.2 Classification supervisée

Nous proposons un schéma itératif où chaque séquence non étiquetée u est affectée en cherchant des concepts compatibles avec u. Un *concept compatible avec u* (cf définition 4.1) est défini comme un concept possédant au moins une classe en extension et partageant un ensemble maximal de blocs avec la séquence non étiquetée. La maximalité est définie ici par rapport à l'inclusion ensembliste. On note C_u l'ensemble des classes compatibles de s, présentes dans les concepts compatibles avec u.

Définition 4.1 (concept compatible avec une séquence u) *Étant donné un treillis de concept $B = \mathscr{B}((L+U+C),A,I)$ et un élément u de U, un concept compatible avec u est un concept tel qu'il existe $c \in C$, $(\{u,c\} \cup X, Y) \in B$, $X \subset (L+U+C)$, et que l'ensemble Y est maximal pour cette propriété (il n'existe pas de concept tel que $(\{u,c\} \cup X', Z)$ avec $Y \subset Z$).*

L'ensemble des classes compatibles $c \in C_u$ représente les classes présentes dans les concepts dont l'extension contient u et c simultanément, avec l'intention la plus spécifique. Les concepts privilégiés à cette étape sont les concepts les plus spécifiques du treillis, où u partage le plus de blocs avec les séquences étiquetées des différentes classes.

A ce stade, l'ensemble C_u peut contenir un grand nombre de classes, en particulier s'il existe un ou plusieurs blocs partagés par l'ensemble des séquences. Toutes les séquences u passant par de tels blocs possèdent alors comme classes compatibles l'ensemble des classes présentes dans l'échantillon. Il est donc important de considérer à la fois le nombre de blocs et de séquences étiquetées impliqués pour effectuer le classement. Nous calculons pour cela pour chaque séquence ses *concepts attribut-compatibles maximaux* et ses *concepts objet-compatibles maximaux* (cf définition 4.2).

Définition 4.2 (concept attribut-compatible et objet-compatible maximaux) *Étant donné un treillis de concepts $B = \mathscr{B}((L+U+C),A,I)$, et une séquence à classer $u \in U$, un concept attribut-compatible et un concept objet-compatible sont des concepts compatibles*

$$BC_{att}(u) = \{(\{u\} \cup X_{att}, Y_{max}) | \exists c \in X_att, c \in C_u\} \text{ et}$$

$$BC_{obj}(u) = \{(\{u\} \cup X_{max}, Y) | \forall c \in X_{att}, c \in X_{max}\} \text{ où,}$$

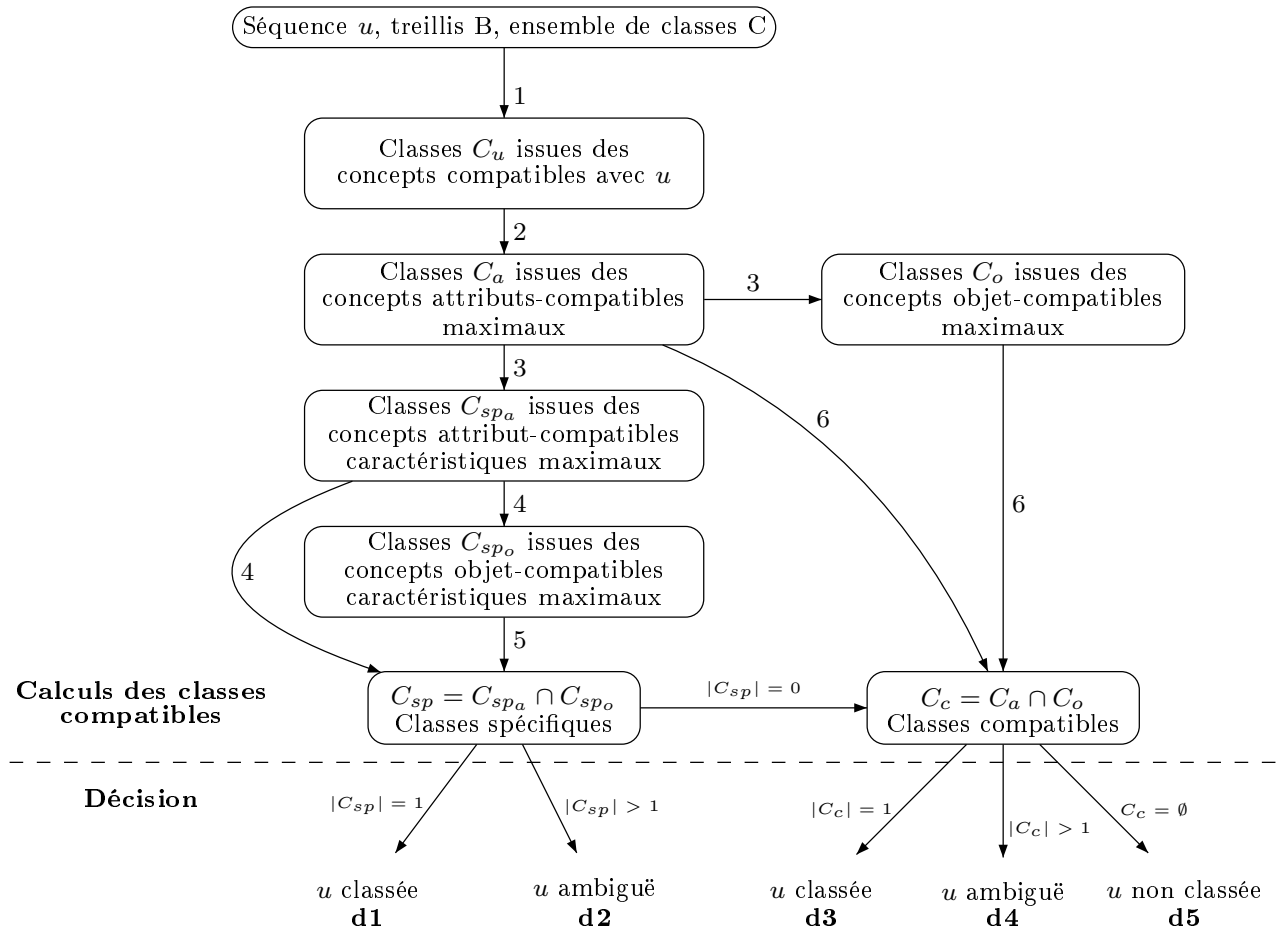


Fig. 4.5 – Détail du processus de classification d'une séquence s au niveau supervisé. Les numéros indiqués sur les arcs définissent l'ordre dans lequel s'effectuent les différentes étapes de calcul.

$$Y_{max} = max\{Y \subset A : (\{u\} \cup X, Y) \in B, X \subset L + U + C\} \ et$$
$$X_{max} = max\{X' \subset L : (\{u\} \cup X, Y) \in B, Y \subset A, X' \subset X\}.$$

Autrement dit, un concept attribut-compatible maximum pour une séquence u *est un concept compatible avec un nombre maximum de blocs. Pour chaque concept attribut-compatible maximum* BC_{att}*, il existe des concepts compatibles tels qu'ils contiennent exactement les classes de* BC_{att}*, les concepts objets-compatibles maximaux sont alors ceux supportés par un nombre maximum de séquences.*

Les ensembles de classes appartenant aux concepts compatibles maximaux d'une séquence u (attribut et objet) sont notés respectivement C_a et C_o, on les appelle des

classes attribut-compatibles maximales (ou objet-compatibles maximales). L'ensemble des classes compatibles maximales est leur intersection $C_c = C_o \cap c \in C_a$.

Une fois cette étape effectuée, on vérifie si certains blocs de A_s sont présents dans les concepts attributs-compatibles maximaux. Si tel est le cas, on construit l'ensemble des concepts attributs-compatibles caractéristiques en conservant uniquement les concepts dans lesquels se trouve un nombre maximal de blocs caractéristiques puis on construit l'ensemble des concepts objet-compatibles caractéristiques correspondant. L'ensemble des classes caractéristiques maximales présentes dans ces concepts sont notés respectivement C_{sp_a} et C_{sp_o}.

L'ensemble des classes dites *spécifiques* C_{sp} pour une séquence non étiquetée u correspond alors à $C_{sp_a} \cap C_{sp_o}$. La procédure de décision découle directement de ces constructions :

d1. S'il n'y a qu'un seul élément c dans C_{sp}, u est classée comme appartenant à la classe c (grâce à la présence de blocs spécifiques à cette classe).

d2. S'il existe plusieurs éléments dans C_{sp}, la séquence est annotée comme ambiguë et car elle partage des blocs spécifiques à plusieurs classes. Toutes les classes possibles sont affichées. Dans ce cas, il n'existe aucune séquence étiquetée partageant simultanément ces blocs.

d3. Si C_{sp} est vide, on cherche les classes compatibles maximales C_c.

d4. S'il y a plusieurs classes compatibles maximales dans C_c, la séquence a une classification ambiguë et toutes les classes possibles sont affichées. Ce cas se produit lorsqu'une séquence ne partage avec les séquences étiquetées que des blocs non discriminants supportant plusieurs familles.

d5. Si aucune classe n'est compatible maximale, c'est qu'aucun concept n'est compatible avec la séquence. Cela signifie soit que la séquence n'a pas de bloc en commun avec une autre séquence et elle reste non classée, soit qu'elle est membre d'une nouvelle famille jamais observée et qu'elle possède des blocs présents uniquement avec des séquences non étiquetées.

4.2.3 Classification non supervisée

En termes de FCA, une nouvelle famille peut être caractérisée comme pour d'autres familles par un concept associé qui rassemble les séquences de cette famille et les blocs qui forment une signature de cette famille. Ces blocs sont caractéristiques des séquences non étiquetées comme c'est le cas pour les concepts spécifiques, mais cette fois c'est une tâche non supervisée puisque l'ensemble des classes N est inconnu.

Ce problème est similaire à celui du *biclustering* [BPP08]. Cependant, l'objectif du *biclustering* consiste en un partitionnement simultané de l'ensemble des objets et des attributs. Dans notre cas, il n'est pas réaliste de s'attendre à une partition des deux ensembles. Les objets (séquences) partagent de nombreux attributs (blocs) et souvent, c'est la façon dont ils sont combinés qui permet de distinguer différents groupes.

La question de regroupement d'objets à partir d'un contexte formel est abordée dans [GNP13] qui propose une procédure en deux étapes où les concepts formels sont étendus

à des concepts approximatifs lors de la première étape, puis fusionnés dans une deuxième étape quand ils se chevauchent suffisamment. Cette approche s'appuie sur le treillis comme nous le faisons dans le but de trouver des groupes, mais il partage les inconvénients du *biclustering* pour notre domaine d'application. Une partition d'objets est utile mais pas nécessaire dans notre cas et en outre, il n'est pas facile de régler les paramètres liés à la méthode pour obtenir des concepts approximatifs significatifs.

Dans [NPG12], l'idée d'utiliser l'ensemble des concepts formels est développée et l'utilisation de seuil n'est plus nécessaire. Au lieu de partir d'un treillis de concepts objets × attributs, les auteurs proposent de considérer le treillis construit sur la relation entre objets et contextes des concepts afin de construire les groupes d'objets. Cela semble une idée intéressante qui pourrait être expérimentée sur le problème de classification des protéines. Cependant, l'interprétation des clusters devient plus difficile et c'est une préoccupation importante pour le biologiste de maîtriser le processus de décision. Un autre aspect connexe de tous ces procédés est leur nature heuristique. L'analyse de concepts formels est une méthode exacte et il semble un peu dommage de perdre cette propriété dans la classification.

Nous avons fondé notre approche sur l'idée d'associer un concept à chaque classe. Nous sommes aussi intéressés par une recherche exacte des concepts sans réglage des paramètres, une exigence qui implique une spécification précise des concepts cibles. La question de décider de l'apparition de nouvelles familles dans N est non triviale en raison de la conjonction de deux difficultés qui doivent être prises en considération :

- Un ensemble donné de séquences participe à un certain nombre de concepts. Un sous-ensemble de concepts doit être extrait, qui couvre l'ensemble de séquences non classées de façon supervisée ;
- L'ensemble des nouvelles familles ne forme pas nécessairement une partition : en effet, une séquence donnée qui a évolué pour obtenir une capacité bifonctionnelle pourrait appartenir à deux familles différentes.

On cherche à résoudre le problème d'optimisation suivant : trouver une couverture optimale des nouvelles familles de séquences par l'ensemble des concepts comprenant de nouveaux blocs caractéristiques seulement présents dans les séquences non étiquetées. L'optimalité dépend de trois critères de priorité décroissante :

1. minimiser le nombre de séquences ambiguës dans un concept pour rester proche d'une partition ;
2. minimiser la taille de N afin de minimiser le nombre de nouvelles familles dans une hypothèse de parcimonie ;
3. maximiser globalement le support des nouvelles familles en termes de nombre de blocs caractéristiques.

Plus formellement, on considère le treillis de concepts $B = \mathscr{B}((L+U+C),A,I)$. L'ensemble des blocs spécifiques à une nouvelle famille est

$$A_{new} = \{a | \forall BC = (X,Y) \in B, a \in Y \Rightarrow X \subseteq U\}$$

et l'ensemble des séquences non classées à l'étape précédente est

$$S_{new} = \{s \in U | \forall BC = (X,Y), s \in X \Rightarrow Y \subseteq A_{new}\}$$

. Le problème est alors de trouver un sous-ensemble

$$B_{new} \subseteq B$$

tel que $\forall s \in S_{new}, \exists BC = (X,Y) \in B_{new}, s \in X$.
Les critères à optimiser sont les suivants, par ordre de priorité décroissante :

1. minimiser $|S_{amb}|$, avec

$$S_{amb} = \{s_i \in S_{new} | \exists BC_1 = (X_1,Y_1), BC_2 = (X_2,Y_2) \in B, BC_1 \neq BC_2, s_i \in X_1, s_i \in X_2\}$$

2. minimiser $|B_{new}|$
3. maximiser A_{use}, avec $A_{use} = \{a \in A_{new} | \exists BC = (X,Y) \in B_{new}, a \in Y\}$

Il est à noter que certaines séquences déjà classées peuvent être associées à une nouvelle famille si elles possèdent un bloc commun avec celles restant non classées après la classification supervisée.

Ces critères sont codés par un ensemble de contraintes logiques dans le cadre de la programmation par ensemble réponse (*Answer Set Programming*) [BET11]. Les concepts optimaux sont produits par un solveur dédié à travers une énumération des solutions admissibles obtenues par une stratégie *conflict-driven constrained* [GKS12]. En pratique, les solutions exactes peuvent être produites en temps raisonnable par cette approche.

4.3 Expérimentation sur des superfamille d'enzymes

Dans un premier temps, nous avons effectué des expérimentations sur la superfamille des haloacide déhalogénases (HAD) pour montrer l'intérêt d'une telle classification. Nous montrons ensuite qu'il est possible de tirer parti de cette classification pour générer des grammaires localement et contextuellement substituables à partir d'un ensemble de séquences annoté et de l'extraction de classes de séquences via l'analyse de concepts formels.

4.3.1 Expérimentation sur des jeux de données connus

La superfamille des haloacides déhalogénases (HAD) est une grande superfamille (120 193 séquences signalées ; http ://pfam.sanger.ac.uk/clan/CL0137) d'enzymes ubiquitaires présentes dans les tous les règnes de la vie. Le nombre de séquences diffère entre les organismes, d'environ vingt pour les bactéries *Escherichia coli* [KPG+06] à 150-200 dans les espèces modèles utilisées en biologie *Arabidopsis thaliana* et *Homo sapiens* [SSG13]. Les HAD sont impliquées dans une grande variété de processus cellulaires et servent de catalyseurs métaboliques prédominant pour l'hydrolyse de l'ester de phosphate [KT94].

Les enzymes de cette superfamille sont liées par leur capacité à former des liaisons covalentes entre enzyme et substrat par l'intermédiaire d'un site acide aspartique conservé. Ces enzymes catalysent le clivage enzymatique, par substitution de nucléophiles, des liaisons carbone-halogène (C-halogène), mais sont également capables d'autres activités d'hydrolyse comme les réactions de phosphatase (CO-P), phosphonatase (C-P) et phosphoglucomutase (CO-P hydrolyse et transfert intramoléculaire de phosphore). La figure 4.6 fournit un exemple de la structure HAD d'une bactérie.

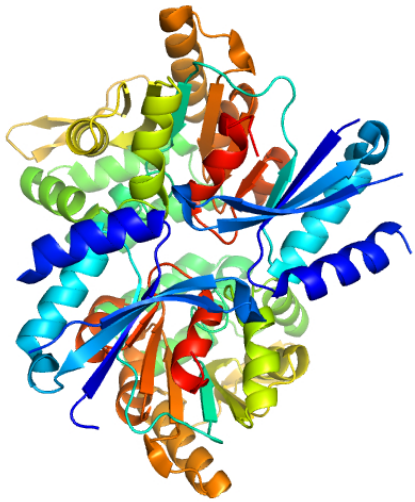


Fig. 4.6 – Structure 3D d'une HAD hydrolase T0658 provenant de *Salmonella enterica*

Tous les membres de la superfamille structurellement caractérisés partagent un domaine α/β-core conservé, appelé repliement "HAD-like" dans la banque de structure SCOP. Les enzymes de cette superfamille fonctionnent généralement sous forme d'homodimères (c'est à dire un complexe composé de deux protéines identiques). Le domaine de base est similaire au repliement de Rossmann avec un feuillet β composé de 6 brins parallèles, et de cinq hélices α. Le repliement typique des HAD phosphatases contient trois signatures structurelles supplémentaires qui permettent à l'enzyme d'adopter des états conformationnels distincts et qui contribuent à la spécificité de leur substrat : les domaines $squiggle$, $flap$ et cap. En effet, la plupart des membres de la superfamille ont un domaine cap, et son site d'incorporation dans la séquence est l'un des paramètres permettant la diversité enzymatique dans la superfamille [BADMA06]. Les déhalogénases ont reçu un intérêt accru dans la dernière décennie car ils ont de potentielles applications industrielles et pharmaceutiques [Jan07].

Pour cette expérience, nous avons travaillé sur les jeux de données suivants :

1. Séquences de plusieurs organismes extraites des suppléments de l'article [BADMA06]. 34 familles, 3 séquences de chaque (102 séquences) ;
2. Séquences de *E. coli* extraites de [KPG+06] 23 séquences ;

3. Séquences de *H. sapiens* extraites de [SSG13] 40 séquences ;
4. Séquences de *A. thaliana* extraites de la base de données TAIR en identifiant les protéines contenant un domaine HAD et des séquences trouvées suite à une analyse bibliographique [BADMA06] 153 séquences, comprenant 23 séquences de fonction inconnue.

Dans toutes les expériences suivantes, le premier ensemble de données est utilisé comme ensemble étiqueté et contient des séquences dont on connaît déjà la classe. Les trois ensembles de données restants ont été utilisés comme des ensembles non étiquetés, et la prédiction de la famille de ces séquences faite par analyse de concepts formels a été comparée à la classification réelle quand elle existe afin d'évaluer la performance de notre analyse. En effet, de nombreuses séquences de *E. coli*, *Homo sapiens* et Arabidopsis thaliana ont été caractérisées biochimiquement et / ou ont été analysées structurellement *in silico*/*in vivo*, ce qui nous donne une connaissance de leur classification réelle.

La figure 4.7 montre le treillis complet obtenu sur le plus petit contexte correspondant à l'expérimentation faite avec les séquences de E.coli comme séquences non étiquetées.

Ce schéma a été tracé à l'aide du logiciel Erca (Eclipse's Relational Concept Analysis [1]) et un étiquetage réduit. Le concept 0 contient tous les blocs et aucune séquence ou classe. Le concept 4 contient toutes les séquences et classes et aucun bloc. Les lignes partant du concept 9 sont légèrement emmêlées avec d'autres et nous avons utilisé une couleur bleue pour mieux les distinguer. Les concepts ayant au moins une séquence à classer dans la figure sont colorés en vert. Ces concepts contiennent l'ensemble des blocs des séquences à classer, un sous-ensemble maximal doit être utilisé pour la classification.

Les résultats d'annotation sont résumés dans la table 4.2.

		E. coli	*H. sapiens*	*A. thaliana*
Classées (%)	**Vrai**	**61**	**65**	**56**
	Faux	9	3	6
Ambiguës (%)	Vrai	17	18	18
	Faux	13	3	8
Non classées (%)	Vrai	0	8	8
	Faux	0	3	5
Total		100	100	100

Tab. 4.2 – Pourcentage des séquences correctement et incorrectement classées par espèce

La ligne «classées» fait référence à des séquences prédites avec une seule classe compatible. La ligne «ambiguës» fait référence à des séquences prédites avec plusieurs classes compatibles. Le classement est supposé être correct (vrai) si une de ces classes compatibles est la bonne. Le pourcentage de séquences correctement attribuées est donné.

[1] https ://code.google.com/p/erca/

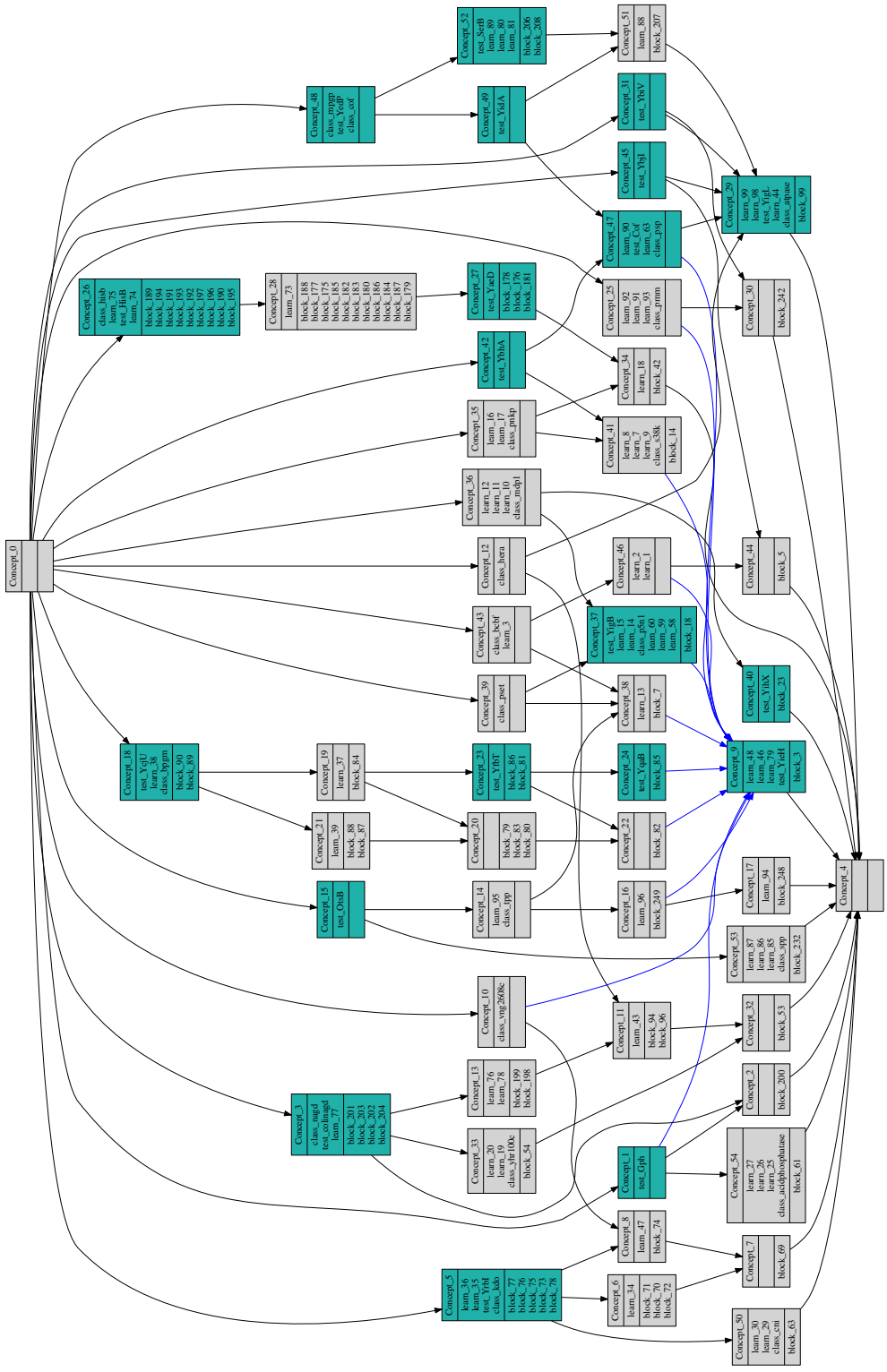
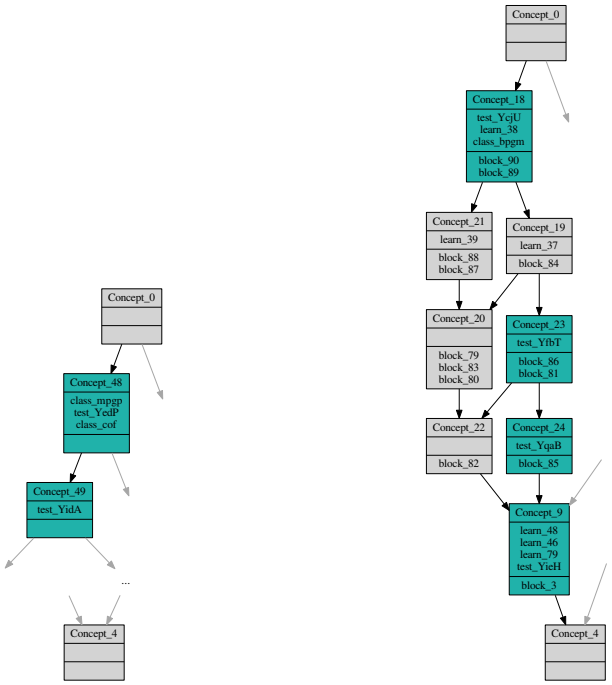


Fig. 4.7 – Diagramme de Hasse du treillis blocs × séquences/classes obtenu sur l'expérimentation avec E. coli comme ensemble de séquences à classer



(a) YedP et YidA sont ambiguës avec deux classes possibles : mpgp et cof

(b) YfbT et YcjU peuvent être classées de façon unique comme bpgm

Fig. 4.8 – Différentes décisions de classification possibles

	H. sapiens	*A. thaliana*
Classées (%)	50	54
Ambiguës (%)	50	21
Non classées (%)	0	25
Total	100	100

Tab. 4.3 – Pourcentage de séquences avec une fonction inconnue dans l'ensemble de test assignée à une ou plusieurs classes

Ces premiers résultats sont encourageants. Plus de 50 % des séquences sont correctement classées dans les 34 familles possibles. Le procédé de détection de nouvelles familles est efficace et les séquences ne faisant pas partie de la superfamille restent non classées.

Pour une partie des séquences à classer, leur vraie classification est en fait inconnue (ensembles de données *H. sapiens* et *A. thaliana*). Pourtant, il est possible de rechercher des affectations possibles de classes. Le tableau 4.3 donne le pourcentage de ces séquences qui pourraient être classées par notre méthode. Elle montre que la plupart de ces séquences inconnues peuvent être affectées à une ou plusieurs classes.

Les pourcentages de séquences appartenant à de nouvelles familles et de séquences non classées sont également donnés. Les séquences non classées sont des séquences qui ne peuvent ni être affectées à une classe connue ni à une nouvelle famille.

Pour les trois ensembles de données, $E.\ coli$, $H.\ sapiens$ et $A.\ thaliana$, nous trouvons 0, 2 et 11 nouvelles sous-familles respectivement.

Dans le cas de $H.\ sapiens$, les séquences prédites comme appartenant à de nouvelles familles sont décrites dans [SSG13]. En fait, ces familles ne sont pas présentes dans l'ensemble étiqueté et sont donc correctement prédites comme nouvelles.

Dans le cas de $A.\ thaliana$, il est difficile de savoir si les nouvelles familles prédites sont réelles en raison du nombre de séquences de fonction inconnue. Cependant, nous avons détecté après une étude bibliographique que 11 séquences non classées semblent avoir été attribuées à tort à la superfamille HAD par la requête de TAIR.

La spécificité de la détection de nouvelles familles a aussi été testée. Pour chaque famille connue dans l'ensemble étiqueté, un nouvel ensemble a été construit qui contient toutes les séquences de l'ensemble initial étiqueté à l'exception des séquences appartenant à cette famille. L'ensemble non étiqueté a été construit avec *E. coli* ainsi que les séquences de la famille sélectionnée (3 séquences). L'objectif était de récupérer toutes les séquences de la famille dans la catégorie $nouvelle\ famille$ par notre méthode. Nous avons calculé le pourcentage de séquences extraites pour toutes les familles. Les résultats sont présentés dans le tableau 4.4. Notez que certaines familles ne sont pas présentes dans *E. coli*. Pour les autres (colonne nouvelle famille + *E. coli*), le nombre de séquences appartenant à la famille est donné entre parenthèses.

nouvelle famille seule	% de séquences retrouvées	nouvelle famille + *E. coli*	% de séquences retrouvées
EYA	100	NagD (+1)	100
SPSC	100	Cof (+6)	44
ATPase	0	PSP (+1)	75
PNKP	100	HisB (+2)	100
deoxy	0	BPGM (+6)	67
s38K	100	Sdt1p (+4)	43
HerA	0	TPP (+1)	100
PMM	100	KDO (+1)	100
Yhr100c	100	MPGP (+1)	67
CNI	67		
Enolase	100		
BCBF	100		
LPIN	100		
PseT	100		
P5N1	100		
AcidPhosphatase	100		
Phosphonatase	100		
VNG2608C	0		
SPP	100		
CNII	100		
MDP1	100		
dehr	0		
Zr25	100		
CTD	100		

Tab. 4.4 – Pourcentage de séquences retrouvées comme une nouvelle famille

Sur les 34 familles présentes dans l'ensemble de départ, cette expérimentation a permis d'en retrouver 27.

4.3.2 Expérimentations sur des données réelles

Afin de tester la robustesse de la méthode sur les espèces qui sont éloignées du point de vue évolutif par rapport aux autres organismes pour lesquels des jeux de tests ont été considérés, nous avons sélectionné l'algue brune *Ectocarpus siliculosus*. La séquence de ce génome a été récemment publiée [MSR+10].

Dans ce génome, 78 séquences ont été annotées comme faisant partie de la superfamille des HAD via le motif de Superfamily « HAD-like superfamily 56784 » sur la base spécialisée des eukariotes [SBA+12] dans laquelle sont répertoriées les séquences du génome de $Ectocarpus$.

Nous avons repris le jeu d'apprentissage utilisé précédemment en remplaçant les séquences à classer par ces 78 séquences. Les résultats de la classification sur ces séquences sont montrés en annexe B.

Nous avons utilisé différents paramètres pour réaliser l'alignement des séquences afin de vérifier la robustesse de la méthode vis à vis de l'alignement. Pour cela, nous avons fait varier un paramètre t visant à définir le seuil de similarité nécessaire pour aligner deux fragments de séquences. Nous avons réalisé trois expériences où $t = \{1,3,5\}$, 1 étant le seuil autorisant le plus d'alignements, 5 étant un seuil très strict mais donnent des résultats plus fiables.

Dans un premier temps, on peut remarquer que beaucoup de séquences ont été classées malgré l'éloignement évolutionnaire d'$Ectocarpus$ par rapport aux espèces modèles. De plus, la plupart des séquences classées grâce à un concept spécifique semblent rester classées, même après variation des paramètres. Ici, les groupes de séquences formant les nouvelles familles, sont très proches d'une expérience à l'autre se qui tend à penser que notre méthode réussit à caractériser la spécificité de chaque famille connue et inconnue.

4.3.3 Apprentissage de grammaire sur une superfamille de séquences

Un des avantages à utiliser l'analyse de concepts formels consiste à pouvoir expliquer la prise de décision dans la classification et pour la détection de nouvelles familles. Il est alors possible de construire des signatures de familles à partir de cette classification.

En effet, bien que l'approche substituabilité soit efficace pour distinguer des motifs au sein d'une même famille, cette approche possède des limites lorsqu'il s'agit de définir la grammaire entière d'une superfamille.

Or, il est plus fréquent d'avoir un bon échantillon caractéristique d'une superfamille que d'une famille, à cause de motifs mieux détectables au sein des séquences.

Cependant, les motifs conservés, présents sur l'ensemble des séquences, brouillent le signal des contextes spécifiques à certaines familles ce qui entraine des substituabilités non souhaitées et des pertes de corrélations.

L'idée est ici d' utiliser l'approche treillis pour trouver les familles présentes (connues ou non) et ainsi sélectionner les contextes intéressants, spécifiques aux familles.

Il suffit ensuite de construire une grammaire par famille en utilisant uniquement les séquences de celle-ci. L'ensemble des autres familles peut alors être vu comme un échantillon négatif servant à éviter la surgénéralisation qu'entraîne la substituabilité à l'intérieur

Algorithm 6: Construction d'une grammaire de superfamille à l'aide de l'analyse de concepts formels

Input: Ensemble séquences S sur l'alphabet Σ des blocs
Input: Ensemble classes $C = C' + unknown$
Input: Ensemble de couples K de $(S \times C)$ K (une étiquette *unknown* signifie une séquence de classe inconnue)
Input: Treillis de concept B issus du plma de S et de K
Output: Grammaire $G = \langle \Sigma, N_K, S_K, P_K \rangle$

$N_K \leftarrow \emptyset, P_K \leftarrow \emptyset$
/* Définition de sous-ensembles de séquences d'une même famille dans S */
$(f, C_2) = classification(B)$
/* C_2 est le nouvel ensemble de classes disponible après classification et f est la fonction renvoyant les classes compatibles à partir d'une séquence s. Si s était déjà annotée, elle revise son annotation. */
$SF = \emptyset$
for $c \in C_2$ **do**
    $S_c = \{s \in S | c \in f(s)\}$
    $SF = SF \cup S_c$
/* Construction d'une grammaire pour chaque famille sur le même alphabet Σ */
for $SF_i \in SF$ **do**
    $G_i \langle \Sigma, N_i, S_i, P_i \rangle = ReGlis(SF_i, \Sigma, k, l)$
    $N_K \leftarrow \{N_K \cup N_i\}$
    $P_K \leftarrow \{P_K \cup P_i\}$
    $P_K \leftarrow \{P_K \cup (S_K \rightarrow S_i)\}$
return $\langle \Sigma, N_K, S_K, P_K \rangle$

de blocs contextes spécifiques à une famille.

Une fois l'ensemble des grammaires apprises sur les différentes familles, on peut construire une *supergrammaire* en ajoutant un axiome S produisant l'union des grammaires de chaque famille, l'alphabet (les blocs) restant le même pour l'ensemble des grammaires puisque tiré d'un même PLMA.

La méthode est expliquée plus formellement dans l'algorithme 6.

Cette grammaire peut ensuite servir à analyser de nouvelles séquences.

Ainsi, si cette grammaire reconnaît une nouvelle séquence, l'arbre de dérivation permettra de connaître la famille à laquelle elle appartient.

Concrètement, si on reprend le schéma de la figure 3.9, cet algorithme remplace l'étape (d).

Un exemple de grammaire obtenu sur la famille des GH16 est donné en annexe C.

4.4 Conclusion de cette approche

Il reste à vérifier expérimentalement la cohérence des résultats obtenus mais les premières expériences semblent montrer un potentiel intéressant.

Nous avons décrit une méthode de classification basée sur un treillis de concepts comprenant à la fois un ensemble d'objets déjà classés et un ensemble d'objets à classer. Elle a été appliquée à des séquences d'enzymes, un groupe de protéines clés impliquées dans de nombreux processus biochimiques et avec un fort potentiel pour la découverte de nouvelles molécules fonctionnelles. Nos résultats sont encourageants et montrent que notre méthode de classification est sensible et spécifique. Plus de la moitié des séquences à classer sont classées correctement par rapport à la connaissance actuelle de 34 familles. De plus, chaque décision de classification peut être clairement expliquée et liée à des séquences connues ou à certaines positions dans la séquence correspondant à des blocs. L'ambiguïté pourrait être encore réduite dans la pratique par la recherche de séquences qui sont par nature ambiguës parce qu'elles sont constitués par exemple de deux fragments de deux protéines de classes différentes. Ces protéines potentielles, que nous appelons chimères, pourraient être récupérées automatiquement lors de la classification.

Un autre aspect de ce travail est le problème de la classification non supervisée pour les objets avec des attributs qui sont caractéristiques des objets à classer. Nous avons proposé un modèle pour résoudre ce problème comme un problème d'optimisation en tenant compte de l'ambiguïté, de la parcimonie (nombre de nouvelles classes nécessaires) et du support (nombre d'attributs).

À notre connaissance, c'est la première fois que cette question est bien formalisée en bio-informatique. Nous avons mis en place toutes les spécifications dans ce document pour pouvoir le résoudre via la programmation par ensembles réponses ($Answer\ Set\ Programming$), une forme de programmation déclarative adaptée aux problèmes combinatoires [BET11]. Une fois toutes les contraintes exprimées en formules logiques, un programme les transforme dans un (grand) ensemble de formules booléennes et un solveur cherche les modèles possibles de cet ensemble (les réponses), qui donnent accès aux solutions du problème initial. Nous avons utilisé le solveur $Clasp$ développé à l'université de Potsdam [GKS12], laboratoire dans lequel nous avons été accueillis pendant 3 mois.

Enfin, nous avons pu tirer parti de cette classification afin d'inférer une grammaire à partir de la connaissance des classes extraites par analyse de concepts formels.

La grammaire ainsi obtenue présente l'avantage de mettre en évidence les corrélations des différents blocs conservés, et présente un aspect hiérarchique lors de la dérivation d'une séquence par cette grammaire, permettant de retrouver à la fois la superfamille et la famille auxquelles elle appartient.

Chapitre 5

Conclusions et perspectives

5.1 Les contributions apportées

En choisissant de produire un nouveau type de signature pour caractériser les familles d'enzymes, cette thèse nous a permis d'aborder un certain nombre de problèmes dans le domaine de l'apprentissage automatique et de la fouille de données. Nous avons pu proposer des contributions originales dans ce cadre.

Tout d'abord, nous avons abordé le problème de caractérisation d'un ensemble de séquences dans le cadre de la théorie des langages. A partir de là, nous avons pu observer les limites en expressivité des techniques existantes et nous tourner vers la représentation d'une famille en tant langage généré par une grammaire algébrique.

Bien que plusieurs techniques permettaient l'inférence de grammaires algébriques, aucune n'avait la faculté de produire des grammaires permettant à la fois d'identifier la structuration des exemples et de générer une classe de langage identifiable à limite. Parmi les techniques existantes, heuristiques et formelles, celles basées sur des principes proche de celui de substituabilité permettent néanmoins d'obtenir des résultats intéressants sur des corpus réels.

Ainsi, nous nous sommes intéressés à l'inférence de la classe des langages algébriques substituables introduite par A. Clarck [CE07] qui semblait posséder de bonnes propriétés d'apprenabilité.

A ce stade, deux problèmes majeurs se sont posés. Dans le premier, ce type de langage ne semblait pas adapté à la représentation des familles d'enzymes. En effet, la grammaire obtenue sur de telles données est trop spécialisée et ne permet pas de reconnaitre de nouvelles séquences appartenant à la classe de la famille d'enzymes traitée. Nous avons alors autorisé le relâchement de la contrainte de substituabilité globale et introduit de nouvelles classes de langages : les langages i,j-localement et k,l-contextuellement substituables. Ces classes nous ont permis de définir de nouveaux principes de généralisation mieux adaptés aux données biologiques. Il a alors été possible d'obtenir des signatures de familles d'enzymes grâce à des grammaires hors-contexte et les premières expériences ont

montrées qu'elles étaient capables de généraliser l'ensemble d'apprentissage à d'autres séquences appartenant à cette famille et non présentes dans l'échantillon.

Le deuxième problème a été d'obtenir une grammaire permettant de déterminer la structuration réelle des exemples qui n'a aucune raison de ressembler à la structure artificielle générée par la forme normale de Chomsky utiliser classiquement pour limiter la combinatoire de l'espace de recherche. Pour cela, nous avons développé l'algorithme ReGliS qui permet d'inférer une grammaire réduite d'un langage substituable à partir d'exemples positifs. La grammaire réduite apprise correspond à la forme qui a été montrée canonique pour les langages substituables la forme canonique apprise dans [Cla14]. Cette approche peut être appliquée aussi bien pour le langage naturel que pour les séquences biologiques et permet d'obtenir une grammaire plus $lisible$ de laquelle il est possible d'extraire la structuration sous-jacente du langage. Pour les enzymes, elle permet de détecter certaines corrélations entre les différentes zones conservées d'une séquence.

Dans un deuxième temps, en côtoyant le monde biologique, nous nous sommes aperçus que le problème même d'obtenir un échantillon d'exemples positifs appartenant à une même famille enzymatique n'était pas trivial.

En effet, au vu du peu d'enzymes caractérisées expérimentalement et du caractère homologue de certaines enzymes de fonctions différentes appartenant à une même superfamille, il est parfois difficile de déterminer la famille d'une enzyme. A l'inverse, repérer l'appartenance d'une séquence à une superfamille semble un problème plus facile dû à la présence des fragments de séquences très conservés. Apprendre la grammaire d'une superfamille directement avec notre algorithme a semblé un problème trop ambitieux. En effet, à cause des motifs de superfamille présents sur la totalité des séquences, le langage ainsi généré présentait une surgénéralisation de la famille d'enzymes à cause de la substituabilité de blocs entre différentes familles.

Pour résoudre ce problème, et en tirant parti des bonnes propriétés qu'offre un alignement local partiel multiple, nous nous sommes alors tournés vers l'analyse de concepts formels afin de classer les séquences d'une superfamille en familles. Pour cela, nous avons implémenté un algorithme grâce à la programmation logique qui permet depuis un ensemble de séquences et d'annotations, de classer des séquences d'une superfamille dans des classes connues ou inconnues en utilisant un procédé de classification supervisé, puis non supervisé si toutes les séquences n'ont pas pu être classées. La classification est basée sur une analyse de concepts formels qui permet de garder la trace des décisions prises et d'expliquer la classification proposée.

Ce procédé peut être utilisé indépendamment de la recherche de signature d'une famille pour classer les séquences d'une superfamille d'un nouvel organisme. Il permet en particulier la détection des nouvelles familles, fréquemment présentes dans les nouveaux organismes non modèles séquencés.

Nous l'avons également utilisé pour créer une grammaire de superfamille générée à partir de l'union des grammaires obtenues pour chaque ensemble de séquences classées dans une famille.

5.2 Perspectives

Utilisation à grande échelle et visualisation

L'une des premières perspectives évidentes serait de procéder à des apprentissages massifs de grammaires de familles et superfamilles d'enzymes afin de construire une banque comparable à celle de Prosite ou de Pfam. L'objectif serait non seulement de procéder à un parsing des banques de données existantes grâce à ces modèles pour trouver des nouveaux membres mais aussi de pouvoir visualiser sur les grammaires les corrélations et dépendances présentes sur la séquence, impliquées ou non dans la fonction.

Cependant, malgré la forme réduite introduite ici, la visualisation difficile d'une grammaire hors-contexte reste une forte limite à sa compréhension. En effet, la lecture de la grammaire obtenue n'est pas évidente pour identifier les zones intéressantes. Pour les protéines, une approche possible serait de s'inspirer de la visualisation d'un arbre de dérivation et de le projeter à la fois sur sa séquence et sa structure afin de mieux visualiser les interactions. Sur cette visualisation il faudrait pouvoir identifier les dérivations alternatives afin de repérer les zones substituables.

Vers les grammaires multiples hors-contexte

Une deuxième perspective intéressante serait de sélectionner les blocs à utiliser en tant que contextes pour la généralisation par substituabilité dans l'inférence produite par ReGLiS. En effet, nous avons simplement proposé dans nos travaux pour caractériser les superfamilles de faire l'union des grammaires de chaque famille, mais il pourrait être intéressant d'autoriser des substituabilités uniquement à l'intérieur de contextes dont les blocs font partie d'un concept spécifique à une famille pour éviter une surgénéralisation du langage de la superfamille.

Dans ce cadre, l'utilisation des grammaires multiples hors-contexte introduite dans [Pol84] pourrait être une idée.

Ce type de grammaire présente plusieurs avantages du fait de sa structure et permettrait de substituer en même temps des groupes de facteurs au lieu d'un seul, ce qui résoudrait le problème de la surgénéralisation lors de l'apprentissage d'une grammaire avec un ensemble de séquences d'une superfamille entière. On peut alors envisager de substituer directement certains concepts mis en avant dans le treillis de classification. De plus, [Yos11] a montré que ces grammaires étaient apprenables grâce a un principe de substituabilité multiple. Il faudrait donc explorer cette piste dans le cadre de la caractérisation d'une superfamille et voir si les concepts peuvent être étendus localement. L'idée serait qu'en partant des exemples de séquences suivantes $maxbycn$ et $pagbhcq$, on substitue directement le groupe de facteur (x,y) par (g,h) dans le contexte (a,b,c) ce qui permet notamment de garder les corrélations entre les éléments a, b et c.

Introduction de connaissances

Une dernière perspective est celle de l'introduction de connaissances *a priori* dans le traitement de la classification des séquences. Cette perspective semble facilement réalisable grâce à la structure même d'un treillis de concept. Les attributs utilisés dans ce manuscrit se limitent aux blocs conservés des séquences mais on peut imaginer y ajouter des informations concernant non seulement la fonction mais aussi la structure ou l'organisme dont est extraite la séquence. Cela permettrait d'avoir une vision et une compréhension encore plus précise de certains blocs présents sur les séquences et de leur fonction (enzymatique, structurale, ...) au sein de celles-ci.

Bibliographie

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, pages 403–410, October 1990.

[AHS09] Adrian K Arakaki, Ying Huang, and Jeffrey Skolnick. EFICAz2 : enzyme function inference by a combined approach enhanced by machine learning. *BMC Bioinformatics*, 10(1) :1, 2009.

[AJH+03] Julie Allouch, Murielle Jam, William Helbert, Tristan Barbeyron, Bernard Kloareg, Bernard Henrissat, and Mirjam Czjzek. The three-dimensional structures of two β-agarases. *Journal of Biological Chemistry*, 278(47) :47171–47180, 2003.

[AM97] Naoki Abe and Hiroshi Mamitsuka. Predicting Protein Secondary Structure Using Stochastic Tree Grammars. *Machine Learning*, 29(2-3) :275–301, 1997.

[AM01] Saïd Abdeddaïm and Burkhard Morgenstern. Speeding Up the DIALIGN Multiple Alignment Program by Using the 'Greedy Alignment of BIOlogical Sequences LIBrary' (GABIOS-LIB). In Olivier Gascuel and Marie-France Sagot, editors, *Computational Biology*, volume 2066 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg, 2001.

[Ang80] Dana Angluin. Inductive inference of formal languages from positive data. *Information and control*, 45(2) :117–135, 1980.

[Ang82] D. Angluin. Inference of Reversible Languages. *J. ACM*, 29(3) :741–765, 1982.

[Ang87] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2) :87–106, 1987.

[AV12] Ulavappa B. Angadi and M. Venkatesulu. Structural SCOP Superfamily Level Classification Using Unsupervised Machine Learning. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 9(2) :601–608, 2012.

[BADMA06] A. Maxwell Burroughs, Karen N. Allen, Debra Dunaway-Mariano, and L. Aravind. Evolutionary Genomics of the HAD Superfamily : Understanding the Structural Adaptations and Catalytic Diversity in a Superfamily of Phosphoesterases and Allied Enzymes. *Journal of Molecular Biology*, 361(5) :1003–1034, 2006.

[Bai00] Amos Bairoch. The ENZYME database in 2000. *Nucleic Acids Research*, 28(1) :304–305, 2000.

[BB01] Pierre Baldi and Soren Brunak. *Bioinformatics : The Machine Learning Approach*. Cambridge : MIT Press, 2nd edition, 2001.

[BBB+09] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble. MEME Suite : tools for motif discovery and searching. *Nucleic Acids Research*, 37(suppl 2) :W202–W208, 2009.

[BCD+04] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl 1) :D138–D141, 2004.

[BET11] Gerhard Brewka, Thomas Eiter, and Miros Truszczynski. Answer Set Programming at a Glance. *Commun. ACM*, 54(12) :92–103, December 2011.

[BHKM06] Markus Brameier, Josien Haan, Andrea Krings, and Robert MacCallum. Automatic discovery of cross-family sequence features associated with protein function. *BMC Bioinformatics*, 7 :16, 2006.

[BJEG98] Alvis Brazma, Inge Jonassen, Ingvar Eidhammer, and David Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of computational biology*, 5(2) :279–305, 1998.

[Blo33] Leonard Bloomfield. *Language*. Henry holt and company, 1933.

[BPP08] Stanislav Busygin, Oleg Prokopyev, and Panos M. Pardalos. Biclustering in Data Mining. *Comput. Oper. Res.*, 35(9) :2964–2987, September 2008.

[BTB00] Christopher Bystroff, Vesteinn Thorsson, and David Baker. HMMSTR : a hidden Markov model for local sequence-structure correlations in proteins. *Journal of molecular biology*, 301(1) :173–190, 2000.

[C+14] UniProt Consortium et al. Activities at the Universal Protein Resource (UniProt). *Nucleic acids research*, 42(D1) :D191–D198, 2014.

[CCGL11] Rafael Carrascosa, François Coste, Matthias Gallé, and Gabriel G. Infante López. The Smallest Grammar Problem as Constituents Choice and Minimal Grammar Parsing. *Algorithms*, 4(4) :262–284, 2011.

[CCR+09] Brandi L Cantarel, Pedro M Coutinho, Corinne Rancurel, Thomas Bernard, Vincent Lombard, and Bernard Henrissat. The Carbohydrate-Active EnZymes database (CAZy) : an expert resource for glycogenomics. *Nucleic acids research*, 37(suppl 1) :D233–D238, 2009.

[CE07] A. Clark and R. Eyraud. Polynomial Identification in the Limit of Substitutable Context-free Languages. *Journal of Machine Learning Research*, 8 :1725–1745, August 2007.

[CGG+14] François Coste, Gaëlle Garet, Agnès Groisillier, Jacques Nicolas, and Thierry Tonon. Automated enzyme classification by formal concept analysis. In *Formal Concept Analysis*, pages 235–250. Springer, 2014.

[CGN12] François Coste, Gaelle Garet, and Jacques Nicolas. Local Substitutability for Sequence Generalization. In Jeffrey Heinz, Colin de la Higuera, and Tim Oates, editors, *ICGI 2012*, volume 21 of *JMLR Workshop and Conference Proceedings*, pages 97–111, Washington, États-Unis, Sep 2012. MIT Press.

[Cho57] Noah Chomsky. *Syntactic Structures*. Mouton & Co., 1957.

[Cho64] Noam Chomsky. Aspects of the theory of syntax. Technical report, DTIC Document, 1964.

[CJS06] David Chiang, Aravind K. Joshi, and David B. Searls. Grammatical Representations of Macromolecular Structure. *Journal of Computational Biology*, 13(5) :1077–1100, 2006.

[CL86a] C. Chothia and A. M. Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, 5(4) :823–826, April 1986.

[CL86b] Cyrus Chothia and Arthur M Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, 5(4) :823, 1986.

[Cla01] Alexander Clark. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 13. Association for Computational Linguistics, 2001.

[Cla03] Alexander Clark. Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 59–66. Association for Computational Linguistics, 2003.

[Cla10a] A. Clark. Distributional Learning of some Context-free Languages with a Minimally Adequate Teacher. In José Sempere and Pedro Garcia, editors, *Grammatical Inference : Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, number 6339 in Lecture Notes in Computer Science, pages 24–37. Springer, Valencia, Spain, September 2010.

[Cla10b] Alexander Clark. Learning Context Free Grammars with the Syntactic Concept Lattice. In José Sempere and Pedro Garcia, editors, *Grammatical Inference : Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference*, number 6339 in Lecture Notes in Computer Science, pages 38–51. Springer, 2010.

[Cla11] Alexander Clark. A Language Theoretic Approach to Syntactic Structure. In Makoto Kanazawa, András Kornai, Marcus Kracht, and Hiroyuki Seki,

editors, *The Mathematics of Language*, volume 6878 of *Lecture Notes in Computer Science*, pages 39–56. Springer Berlin Heidelberg, 2011.

[Cla14] Alexander Clark. Learning Trees from Strings : A Strong Learning Algorithm for some Context-Free Grammars. *Journal of Machine Learning Research*, 14 :3537–3559, 2014.

[CRA76] Craig M Cook, Azriel Rosenfeld, and Alan R Aronson. Grammatical inference by hill climbing. *Information Sciences*, 10(2) :59–80, 1976.

[CRCFK03] Clotilde Claudel-Renard, Claude Chevalet, Thomas Faraut, and Daniel Kahn. Enzyme-specific profiles for genome annotation : PRIAM. *Nucleic acids research*, 31(22) :6633–6639, 2003.

[CV03] Nick Chater and Paul Vitányi. Simplicity : A unifying principle in cognitive science ? *Trends in cognitive sciences*, 7(1) :19–22, 2003.

[DA+00] Laurent Duret, Saïd Abdeddaim, et al. Multiple alignment for structural, functional, or phylogenetic analyses of homologous sequences. *Bioinformatics : Sequence, Structure, and Databanks*, pages 51–76, 2000.

[Day73] Margaret Oakley Dayhoff. *Atlas of Protein Sequence and Structure : Supplement No. 1; Edited [by] MO Dayhoff*. National Biomedical Research Foundation, 1973.

[dKK+97] Marie desJardins, Peter D. Karp, Markus Krummenacker, Thomas J. Lee, and Christos A. Ouzounis. Prediction of Enzyme Classification from Protein Sequence without the Use of Sequence Similarity. In Terry Gaasterland, Peter D. Karp, Kevin Karplus, Christos A. Ouzounis, Chris Sander, and Alfonso Valencia, editors, *ISMB*, pages 92–99. AAAI, 1997.

[dlH97] Colin de la Higuera. Characteristic Sets for Polynomial Grammatical Inference. *Machine Learning*, 27(2) :125–138, 1997.

[DLT02] François Denis, Aurélien Lemay, and Alain Terlutte. Some classes of regular languages identifiable in the limit from positive data. In *Grammatical Inference : algorithms and applications*, pages 63–76. Springer, 2002.

[DMV94] P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference ? In *In Proceedings of the Second International Colloquium on Grammatical Inference (ICGI'94*, pages 25–37. Springer Verlag, 1994.

[DN09] W. Dyrka and J.-C. Nebel. A stochastic context free grammar based framework for analysis of protein sequences. *BMC Bioinformatics*, 10(1) :323+, October 2009.

[DV00] Damien Devos and Alfonso Valencia. Practical limits of function prediction. *Proteins : Structure, Function, and Bioinformatics*, 41(1) :98–107, 2000.

[Edd98] S. Eddy. Hmmer user's guide : biological sequence analysis using prole hidden markov models, 1998.

[EPS+05] Roman Eisner, Brett Poulin, Duane Szafron, Paul Lu, and Russell Greiner. Improving Protein Function Prediction Using the Hierarchical Structure of the Gene Ontology. In *CIBCB*, pages 354–363. IEEE, 2005.

[Eyr06] Rémi Eyraud. Inférence grammaticale de langages hors-contextes. 2006.

[FBC14] Naomi K. Fox, Steven E. Brenner, and John-Marc Chandonia. SCOPe : Structural Classification of Proteins-extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1) :D304–D309, 2014.

[Fre03] Daniel Fredouille. *Inférence d'automates finis non déterministes par gestion de l'ambiguïté, en vue d'applications en bioinformatique*. PhD thesis, Rennes1, 2003.

[FW03] Justin C Fay and Chung-I Wu. Sequence divergence, functional constraint, and selection in protein evolution. *Annual review of genomics and human genetics*, 4(1) :213–235, 2003.

[GB00] John A Gerlt and Patricia C Babbitt. Can sequence determine function. *Genome Biol*, 1(5) :1–10, 2000.

[GD95] Diana F Gordon and Marie Desjardins. Evaluation and selection of biases in machine learning. *Machine Learning*, 20(1-2) :5–22, 1995.

[GK10] M. Y. Galperin and E. V. Koonin. From complete genome sequence to 'complete' understanding ? *Trends in Biotechnology*, 28(8) :398–406, 2010.

[GKS12] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven answer set solving : From theory to practice. *Artif. Intell.*, 187 :52–89, 2012.

[GME87] Michael Gribskov, Andrew D McLachlan, and David Eisenberg. Profile analysis : detection of distantly related proteins. *Proceedings of the National Academy of Sciences*, 84(13) :4355–4358, 1987.

[GNP13] Bruno Gaume, Emmanuel Navarro, and Henri Prade. Clustering bipartite graphs in terms of approximate formal concepts and sub-contexts. *International Journal of Computational Intelligence Systems*, 6(6) :1125–1142, 2013.

[Gol67] E Mark Gold. Language identification in the limit. *Information and control*, 10(5) :447–474, 1967.

[Gol78] E.M. Gold. Complexity of Automaton Identification from Given Data. *Information and Control*, 37(3) :302–320, 1978.

[Grü94] Peter Grünwald. A Minimum Description Length Approach to Grammar Inference. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language, volume 1004 of Lecture Notes in AI*, pages 203–216. Springer Verlag, 1994.

[GSSV94] Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated mutations and residue contacts in proteins. *Proteins : Structure, Function, and Bioinformatics*, 18(4) :309–317, 1994.

[Gu03] Xun Gu. Functional Divergence in Protein (Family) Sequence Evolution. *Genetica*, 118(2-3) :133, 2003.

[GV90] P. Garcia and E. Vidal. Inference of k-Testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(9) :920–925, 1990.

[GVC87] P. Garcia, E. Vidal, and F. Casacuberta. Local Languages, the Successor Method, and a Step Towards a General Methodology for the Inference of Regular Grammars. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(6) :841–845, nov. 1987.

[GVO90] P. Garcia, E. Vidal, and J. Oncina. Learning Locally Testable Languages in the Strict Sense. In *First int. workshop on Algorithmic Learning theory, ALT'90*, pages 325–338, 1990.

[HAB+07] Gemma L. Holliday, Daniel E. Almonacid, Gail J. Bartlett, Noel M. O'Boyle, James W. Torrance, Peter Murray-Rust, John B. O. Mitchell, and Janet M. Thornton. MACiE (Mechanism, Annotation and Classification in Enzymes) : novel tools for searching catalytic mechanisms. *Nucleic Acids Research*, 35(suppl 1) :D515–D520, 2007.

[Har54] Z. S. Harris. Distributional Structure. *Word*, (23) :146–162, 1954.

[HBB+06] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S. Langendijk-genevaux, Marco Pagni, and Christian J. A. Sigrist. The prosite database. *Nucleic Acids Res*, 34 :227–230, 2006.

[HCB+10] Jan-Hendrik Hehemann, Gaëlle Correc, Tristan Barbeyron, William Helbert, Mirjam Czjzek, and Gurvan Michel. Transfer of carbohydrate-active enzymes from marine bacteria to Japanese gut microbiota. *Nature*, 464(7290) :908–912, 2010.

[Hea87] T. Head. Formal language theory and DNA : An analysis of the generative capacity of specific recombinant behaviours. *Bulletin of Mathematical Biology*, 49(6) :737–759, 1987.

[HH92] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22) :10915–10919, 1992.

[HHAP95] S. Henikoff, J. G. Henikoff, W. J. Alford, and S. Pietrokovski. Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene*, 163, October 1995.

[HJM+12] S. Hunter, P. Jones, A. Mitchell, R. Apweiler, T. K. Attwood, A. Bateman, T. Bernard, D. Binns, P. Bork, S. Burge, E. de Castro, P. Coggill, M. Corbett, U. Das, L. Daugherty, L. Duquenne, R. D. Finn, M. Fraser, J. Gough, D. Haft, N. Hulo, D. Kahn, E. Kelly, I. Letunic, D. Lonsdale, R. Lopez, M. Madera, J. Maslen, C. McAnulla, J. McDowall, C. McMenamin, H. Mi, P. Mutowo-Muellenet, N. Mulder, D. Natale, C. Orengo, S. Pesseat, M. Punta, A. F. Quinn, C. Rivoire, A. Sangrador-Vegas, J. D. Selengut, C. J. A. Sigrist, M. Scheremetjew, J. Tate, M. Thimmajanarthanan, P. D. Thomas, C. H. Wu, C. Yeats, and S. Yong. InterPro in 2011 :

new developments in the family and domain prediction database. *Nucleic Acids Research*, 40(D1) :D306–D312, 2012.

[IY13] M. Ikeda and A. Yamamoto. Classification by Selecting Plausible Formal Concepts in a Concept Lattice. In *Workshop on Formal Concept Analysis meets Information Retrieval (FCAIR2013)*, pages 22–35, 2013.

[Jan07] Dick B. Janssen. Biocatalysis by Dehalogenating Enzymes. In Sima Sariaslanl Allen I. Laskin and Geoffrey M. Gadd, editors, , volume 61 of *Advances in Applied Microbiology*, pages 233–252. Academic Press, 2007.

[JHH96] Inge Jonassen, Carsten Helgesen, and Desmond Higgins. Scoring Function for Pattern Discovery Programs Taking Into Account Sequence Diversity, 1996.

[Ker08] G. Kerbellec. *Apprentissage d'automates modélisant des familles de séquences protéiques*. PhD thesis, Université Rennes 1, 2008.

[KG02] Lisa N Kinch and Nick V Grishin. Evolution of protein structures and functions. *Current opinion in structural biology*, 12(3) :400–408, 2002.

[KJ58] DE Koshland Jr. Application of a theory of enzyme specificity to protein synthesis. *Proceedings of the National Academy of Sciences of the United States of America*, 44(2) :98, 1958.

[KMT97] Takeshi Koshiba, Erkki Mäkinen, and Yuji Takada. Learning deterministic even linear languages from positive examples. *Theoretical Computer Science*, 185(1) :63–79, 1997.

[Kov07] L. Kovacs. Generating decision tree from lattice for classification. In *7th International Conference on Applied Informatics*, volume 2, pages 377–384, 2007.

[KPG+06] Ekaterina Kuznetsova, Michael Proudfoot, Claudio F. Gonzalez, Greg Brown, Marina V. Omelchenko, Ivan Borozan, Liran Carmel, Yuri I. Wolf, Hirotada Mori, Alexei V. Savchenko, Cheryl H. Arrowsmith, Eugene V. Koonin, Aled M. Edwards, and Alexander F. Yakunin. Genome-wide Analysis of Substrate Specificities of the *Escherichia coli* Haloacid Dehalogenase-like Phosphatase Family. *Journal of Biological Chemistry*, 281(47) :36149–36161, 2006.

[KT94] Eugene V. Koonin and Roman L. Tatusov. Computer Analysis of Bacterial Haloacid Dehalogenases Defines a Large Superfamily of Hydrolases with Diverse Specificity : Application of an Iterative Approach to Database Search. *Journal of Molecular Biology*, 244(1) :125–132, 1994.

[L+91] Thomas F Lee et al. *The human genome project. Cracking the genetic code of life.* Plenum Press, 1991.

[Lan92] Kevin Lang. Random dfa's can be approximately learned from sparse uniform examples. In *In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 45–52. ACM Press, 1992.

[Ler05] Aurélien Leroux. *Inférence grammaticale sur des alphabets ordonnés : Application à la découverte de motifs dans des familles de protéines.* PhD thesis, Université Rennes 1, 2005.

[LNL95] Jun S Liu, Andrew F Neuwald, and Charles E Lawrence. Bayesian models for multiple local sequence alignment and gibbs sampling strategies. *Journal of the American Statistical Association*, 90(432) :1156–1170, 1995.

[MHK+99] Hans-Werner Mewes, Klaus Heumann, Andreas Kaps, K Mayer, Friedhelm Pfeiffer, S Stocker, and Dmitrij Frishman. Mips : a database for genomes and protein sequences. *Nucleic acids research*, 27(1) :44–48, 1999.

[Mit80] Tom M Mitchell. *The need for biases in learning generalizations.* Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ., 1980.

[Mor99] Burkhard Morgenstern. Dialign 2 : improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15(3) :211–218, 1999.

[Mou08] David W Mount. Comparison of the pam and blosum amino acid substitution matrices. *Cold Spring Harbor Protocols*, 2008(6) :pdb–ip59, 2008.

[MSR+10] J. Mark Cock, L. Sterck, P. Rouzé, D. Scornet, A. Allen, G. Amoutzias, V. Anthouard, F. Artiguenave, J. Aury, and J. Badger. The Ectocarpus genome and the independent evolution of multicellularity in brown algae. *Nature*, (7298) :617–621, 2010.

[NPG12] Emmanuel Navarro, Henri Prade, and Bruno Gaume. Clustering Sets of Objects Using Concepts-Objects Bipartite Graphs. In Eyke Hüllermeier, Sebastian Link, Thomas Fober, and Bernhard Seeger, editors, *SUM*, volume 7520 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 2012.

[NW70] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3) :443–453, 1970.

[OG92] J. Oncina and P. Garcia. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, pages 49–61, 1992.

[Ogd68] William Ogden. A helpful result for proving inherent ambiguity. *Theory of Computing Systems*, 2(3) :191–194, 1968.

[OMJ+97] C.A. Orengo, A.D. Michie, D.T. Jones, M.B. Swindells, and J.M. Thornton. CATH : A Hierarchic Classification of Protein Domain Structures. *Structure*, (5) :1093–1108, 1997. Using secondary structures to measure the geometry of a protein.

[Pit89] L. Pitt. Inductive Inference, DFAs, and Computational Complexity. In K.P. Jantke, editor, *Proceedings of International Workshop on Analogical and Inductive Inference (AII)*, volume 397 of *Lecture Notes in Computer Science*, pages 18–44. Springer-Verlag, Berlin, 1989.

[PLC08] Piedachu Peris, Damián López, and Marcelino Campos. IgTM : An algorithm to predict transmembrane domains and topology in proteins. *BMC Bioinformatics*, 9, 2008.

[PLCS06] P. Peris, D. López, M. Campos, and J. M. Sempere. Protein Motif Prediction by Grammatical Inference. In Sakakibara et al. [SKS+06], pages 175–187.

[Pol84] Carl Jesse Pollard. Generalized phrase structure grammars, head grammars and natural language. *Ph. D. dissertation, Stanford University*, 1984.

[PP33] Anselme Payen and Jean-François Persoz. Mémoire sur la diastase, les principaux produits de ses réactions, et leurs applications aux arts industriels. *Ann. chim. phys*, 53(2) :73–92, 1833.

[PPK+04] Georgios Petasis, Georgios Paliouras, Vangelis Karkaletsis, Constantine Halatsis, and Constantine D. Spyropoulos. E-GRIDS : Computationally Efficient Grammatical Inference from Positive Examples. *GRAMMARS*, 7 :69–110, 2004. Technical Report referenced in the paper : http ://www.ellogon.org/petasis/bibliography/GRAMMARS/GRAMMARS2004-SpecialIssue-Petasis-TechnicalReport.pdf.

[RCO+13] Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, Gaurav Pandey, Jeffrey M Yunes, Ameet S Talwalkar, Susanna Repo, Michael L Souza, Damiano Piovesan, Rita Casadio, Zheng Wang, Jianlin Cheng, Hai Fang, Julian Gough, Patrik Koskinen, Petri Toronen, Jussi Nokso-Koivisto, Liisa Holm, Domenico Cozzetto, Daniel W A Buchan, Kevin Bryson, David T Jones, Bhakti Limaye, Harshal Inamdar, Avik Datta, Sunitha K Manjari, Rajendra Joshi, Meghana Chitale, Daisuke Kihara, Andreas M Lisewski, Serkan Erdin, Eric Venner, Olivier Lichtarge, Robert Rentzsch, Haixuan Yang, Alfonso E Romero, Prajwal Bhat, Alberto Paccanaro, Tobias Hamp, Rebecca Kaszner, Stefan Seemayer, Esmeralda Vicedo, Christian Schaefer, Dominik Achten, Florian Auer, Ariane Boehm, Tatjana Braun, Maximilian Hecht, Mark Heron, Peter Honigschmid, Thomas A Hopf, Stefanie Kaufmann, Michael Kiening, Denis Krompass, Cedric Landerer, Yannick Mahlich, Manfred Roos, Jari Bjorne, Tapio Salakoski, Andrew Wong, Hagit Shatkay, Fanny Gatzmann, Ingolf Sommer, Mark N Wass, Michael J E Sternberg, Nives Skunca, Fran Supek, Matko Bosnjak, Pance Panov, Saso Dzeroski, Tomislav Smuc, Yiannis A I Kourmpetis, Aalt D J van Dijk, Cajo J F ter Braak, Yuanpeng Zhou, Qingtian Gong, Xinran Dong, Weidong Tian, Marco Falda, Paolo Fontana, Enrico Lavezzo, Barbara Di Camillo, Stefano Toppo, Liang Lan, Nemanja Djuric, Yuhong Guo, Slobodan Vucetic, Amos Bairoch, Michal Linial, Patricia C Babbitt, Steven E Brenner, Christine Orengo, Burkhard Rost, Sean D Mooney, and Iddo Friedberg. A large-scale evaluation of computational protein function prediction. *Nat Meth*, advance online publication, January 2013.

[Ris78] J. Rissanen. Modeling by the shortest data description. *Automation*, 14 :465–471, 1978.

[RZM+04] Andreas Ruepp, Alfred Zollner, Dieter Maier, Kaj Albermann, Jean Hani, Martin Mokrejs, Igor Tetko, Ulrich Güldener, Gertrud Mannhaupt, Martin Münsterkötter, et al. The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic acids research*, 32(18) :5539–5545, 2004.

[Sah95] Mehran Sahami. Learning Classification Rules Using Lattices. In Nada Lavrac and Stefan Wrobel, editors, *ECML*, volume 912 of *Lecture Notes in Computer Science*, pages 343–346. Springer, 1995.

[SB02] Ismael Salvador and José-Miguel Benedí. RNA Modeling by Combining Stochastic Context-Free Grammars and n-Gram Models. *IJPRAI*, 16(3) :309–315, 2002.

[SBA+12] Lieven Sterck, Kenny Billiau, Thomas Abeel, Pierre Rouze, and Yves Van de Peer. ORCAE : online resource for community annotation of eukaryotes. *Nat Meth*, 9(11) :1041, November 2012.

[SBU+94] Yasubumi Sakakibara, Michael Brown, Rebecca C. Underwood, I. Saira Mian, and David Haussler. Stochastic Context-Free Grammars for Modeling RN. In *HICSS (5)*, pages 284–294, 1994.

[SC75] Fred Sanger and Alan R Coulson. A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *Journal of molecular biology*, 94(3) :441–448, 1975.

[SC07] Hong-Bin Shen and Kuo-Chen Chou. EzyPred : a top-down approach for predicting enzyme functional classes and subclasses. *Biochem. Biophys. Res. Commun.*, 364(1) :53–9, December 2007.

[SCD+13] Ian Sillitoe, Alison L. Cuff, Benoit H. Dessailly, Natalie L. Dawson, Nicholas Furnham, David Lee, Jonathan G. Lees, Tony E. Lewis, Romain A. Studer, Robert Rentzsch, Corin Yeats, Janet M. Thornton, and Christine A. Orengo. New functional families (FunFams) in CATH to improve the mapping of conserved functional sites to 3D structures. *Nucleic Acids Research*, 41(D1) :D490–D498, 2013.

[Sch95] Hinrich Schütze. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148. Morgan Kaufmann Publishers Inc., 1995.

[SCP+13] Ida Schomburg, Antje Chang, Sandra Placzek, Carola Söhngen, Michael Rother, Maren Lang, and Cornelia Munaretto. BRENDA in 2013 : integrated reactions, kinetic data, enzyme function data, improved disease classification : new options and contents in BRENDA. *Nucleic Acids Research*, 41 :D764–72, January 2013.

[Sea02] David B. Searls. The language of genes. *Nature*, 420 :211–217, 2002.

[SH08] Sarah M Sullivan and Todd Holyoak. Enzymes with lid-gated active sites must operate by an induced fit mechanism instead of conformational selection. *Proceedings of the National Academy of Sciences*, 105(37) :13829–13834, 2008.

[SHRE05] Z. Solan, D. Horn, E. Ruppin, and S. Edelman. Unsupervised learning of natural languages. *PNAS*, 102(33) :11629–11634, 2005.

[SKS+06] Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita, editors. *Grammatical Inference : Algorithms and Applications, 8th International Colloquium, ICGI 2006, Tokyo, Japan, September 20-22, 2006, Proceedings*, volume 4201 of *Lecture Notes in Computer Science*. Springer, 2006.

[SS90] Thomas D Schneider and R Michael Stephens. Sequence logos : a new way to display consensus sequences. *Nucleic acids research*, 18(20) :6097–6100, 1990.

[SSG13] Annegrit Seifried, Jörg Schultz, and Antje Gohla. Human HAD phosphatases : structure, mechanism, and roles in health and disease. *FEBS Journal*, 280(2) :549–571, 2013.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, pages 195–197, March 1981.

[SY09] Umar Syed and Golan Yona. *Enzyme Function Prediction with Interpretable Models*. Springer, 2009.

[TB00] Keith F. Tipton and Sinéad Boyce. History of the enzyme nomenclature system. *Bioinformatics*, 16(1) :34–40, 2000.

[THG94] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, (22) :4673–4680, November 1994.

[Val84] Les Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11) :1134–1142, 1984.

[Vap92] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[vZ00] M. van Zaanen. ABL : Alignment-Based Learning. In *COLING 18*, pages 961–967, 2000.

[WC+53] James D Watson, Francis HC Crick, et al. Molecular structure of nucleic acids. *Nature*, 171(4356) :737–738, 1953.

[Web64] E. C. Webb. The nomenclature of multiple enzyme forms. *Experientia*, 20(10) :592, 1964.

[Web93] Edwin C Webb. Enzyme nomenclature : a personal retrospective. *The FASEB Journal*, 7(12) :1192–1194, 1993.

[Wil82] Rudolf Wille. Restructuring lattice theory : An approach based on hierarchies of concepts. In *Ordered Sets*, pages 445–470. Springer, 1982.

[WKG00] Cyrus A Wilson, Julia Kreychman, and Mark Gerstein. Assessing annotation transfer for genomics : quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores. *Journal of molecular biology*, 297(1) :233–249, 2000.

[WLQ11] Junhong Wang, Jiye Liang, and Yuhua Qian. Closed-Label Concept Lattice Based Rule Extraction Approach. In De-Shuang Huang, Yong Gan, Prashan Premaratne, and Kyungsook Han, editors, *ICIC (3)*, volume 6840 of *Lecture Notes in Computer Science*, pages 690–698. Springer, 2011.

[Wol80] J Gerard Wolfp. Language acquisition and the discovery of phrase structure. *Language and Speech*, 23(3) :255–269, 1980.

[WP99] Todd C. Wood and William R. Pearson. Evolution of Protein Sequences and Structures. *Journal of Molecular Biology*, 291(4) :977–995, August 1999.

[WYD10] Yong-Cui Wang, Zhi-Xia Yang, and Nai-Yang Deng. SVM-based Method for Predicting Enzyme Function in a Hierarchical Context. In *The Fourth International Conference on Computational Systems Biology (ISB2010)*, pages 119–127, 2010.

[YIK94] T. Yokomori, N. Ishida, and S. Kobayashi. Learning Local Languages and its Application to Protein α-Chain Identification. In *HICSS (5)*, pages 113–122, 1994.

[Yos08] R. Yoshinaka. Identification in the Limit of (k,l)-Substitutable Context-Free Languages. In *Proceedings of the 9th international colloquium conference on Grammatical inference : theoretical results and applications*, ICGI'08, pages 266–279, 2008.

[Yos11] Ryo Yoshinaka. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412(19) :1821–1831, 2011.

[ZCBZ11] J. Zhang, R. Chiodini, A. Badr, and G. Zhang. The impact of next-generation sequencing on genomics. *Journal of Genetics and Genomics*, 38(3) :95–109, 2011.

Annexes

A. Alignement partiel local partiel de la superfamille des GH16

Sur l'alignement de la figure A.1, les chemins représentés par une suite d'arc représente une séquence et les blocs sont représentés par des rectangles comprenant certaines positions des séquences.

La couleur des séquences représente leur appartenance à une classe donnée définie comme suit :

- Agarase (EC. 3.2.1.81) : vert
- Kappa carraghenase (EC. 3.2.1.83) : orange
- Keratan sulphate hydrolase (EC. 3.2.1.103) : jaune
- Xyloglucanase (EC. 3.2.1.151) : mauve
- Laminarinase (EC. 3.2.1.39) : rouge
- Lichenase (EC. 3.2.1.73) : bleu

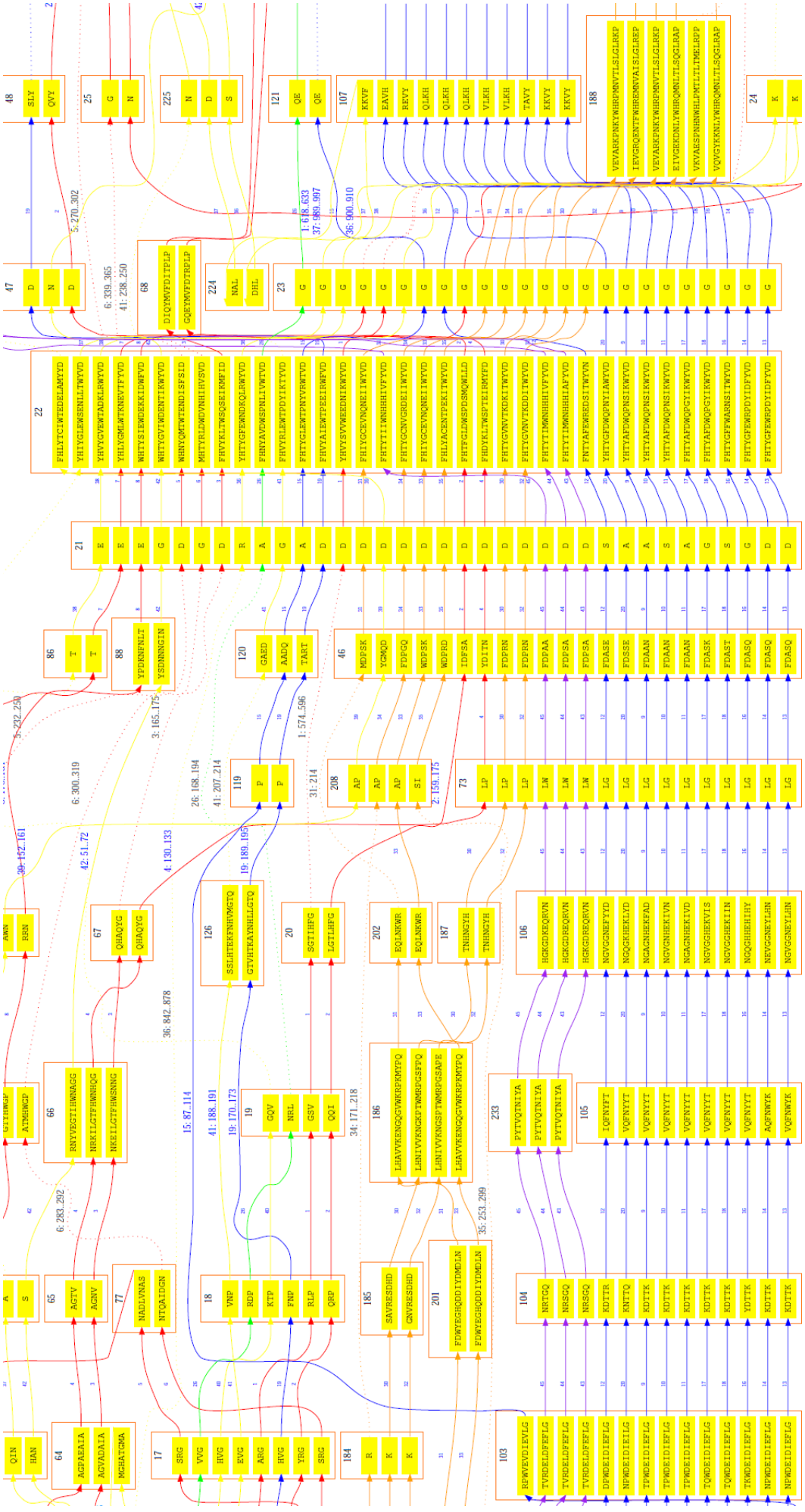


Fig. A.1 – Extrait d'un alignement multiple local et partiel de séquences appartenant à la superfamille des GH16

B. Résultats de classement obtenus sur les séquences HAD d'Ectocarpus

TAB. 1: Classification des séquence HAD d'*Ectocarpus* dans les différentes familles, le jeu de test a été lancé avec 3 parametres différents. Pour chaque séquence, chaque ligne correspond à un parametre (t=1, t=3 et t=5). Les colonnes correspondent respectivement au nom de la séquence, à son statut de classification, aux familles dans lesquelles elle a été classée et éventuellement les sous-familles définies en table2 auxquelles elles ont été ratachées par la classification non supervisée

Séquence	Statut	Familles	Nouvelles familles
Esi0015_0161	classified ambiguous ambiguous	bpgm bpgm cof pset cof	1_31
Esi0004_0179	bestclassified ambiguous unclassified	tpp bcbf cof pset tpp pmm psp s38k	
Esi0207_0008	ambiguous unclassified unclassified	hera atpase psp mpgp cof	
Esi0019_0077	bestclassified bestclassified ambiguous	mdp1 mdp1 pmm mdp1 p5n1	
Esi0017_0035	bestclassified unclassified unclassified	pnkp	
Esi0167_0081	unclassified unclassified unclassified		
Esi0146_0063	bestclassified bestclassified bestclassified	ctd ctd ctd	
Esi0004_0164	bestclassified bestclassified bestclassified	ctd ctd ctd	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0018_0096	bestclassified classified classified	hisb hisb hisb	
Esi0028_0111	unclassified unclassified unclassified		
Esi0237_0016	ambiguous ambiguous bestclassified	atpase hera atpase hera atpase	
Esi0268_0033	bestclassified bestclassified unclassified	atpase atpase	
Esi0180_0054	unclassified classified unclassified	acidphosphatase	
Esi0040_0046	bestclassified ambiguous classified	atpase mpgp cof mpgp	
Esi0054_0026	ambiguous ambiguous bestclassified	atpase hera atpase hera atpase	
Esi0009_0142	bestclassified bestclassified bestclassified	atpase spsc spsc	 1_53 2_37
Esi0287_0019	unclassified unclassified unclassified		
Esi0297_0017	unclassified unclassified unclassified		
Esi0028_0056	ambiguous ambiguous bestclassified	atpase hera atpase hera spsc	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0023_0054	bestclassified unclassified unclassified	atpase	0_3 1_53,1_31 2_37
Esi0129_0048	unclassified unclassified unclassified		
Esi0002_0052	bestclassified bestclassified bestclassified	ctd ctd ctd	
Esi0130_0079	classified ambiguous classified	nagd nagd hera bpgm	
Esi0458_0004	bestclassified bestclassified unclassified	ctd ctd	
Esi0291_0032	bestclassified unclassified unclassified	pmm	
Esi0009_0037	bestclassified bestclassified unclassified	atpase spsc	 1_53 2_37
Esi0020_0166	ambiguous ambiguous unclassified	hera psp mpgp cof mpgp cof	
Esi0169_0029	ambiguous unclassified unclassified	nagd yhr100c hera	
Esi0010_0099	bestclassified bestclassified classified	tpp tpp tpp	
Esi0000_0471	bestclassified bestclassified unclassified	ctd spsc	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0263_0021	bestclassified ambiguous classified	atpase mpgp cof mpgp	
Esi0309_0026	ambiguous ambiguous classified	hera bpgm pnkp bpgm vng2608c cof bpgm	
Esi0116_0053	bestclassified bestclassified bestclassified	ctd ctd spsc	
Esi0047_0101	ambiguous unclassified bestclassified	atpase hera atpase	
Esi0003_0077	ambiguous bestclassified unclassified	hera nagd pset p5n1 sdt1p mdp1 spsc	
Esi0064_0072	bestclassified bestclassified bestclassified	tpp spsc spsc	1_31
Esi0145_0068	ambiguous ambiguous bestclassified	atpase hera atpase hera atpase	
Esi0387_0003	unclassified bestclassified unclassified	 spsc	
Esi0095_0017	unclassified unclassified unclassified		
Esi0291_0033	unclassified unclassified unclassified		
Esi0314_0025	ambiguous unclassified unclassified	hera zr25 bpgm	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0160_0053	bestclassified bestclassified unclassified	pnkp pnkp	
Esi0023_0124	bestclassified classified classified	zr25 zr25 zr25	
Esi0148_0035	ambiguous unclassified unclassified	hera psp mpgp cof	
Esi0048_0016	unclassified bestclassified bestclassified	 spsc spsc	0_3
Esi0243_0024	bestclassified unclassified unclassified	tpp	0_3 1_31
Esi0168_0050	bestclassified ambiguous classified	atpase mpgp cof mpgp	
Esi0039_0126	bestclassified bestclassified bestclassified	tpp spsc spsc	0_3
Esi0206_0037	bestclassified ambiguous unclassified	spp mpgp cof	
Esi0000_0445	ambiguous unclassified unclassified	hera nagd pset p5n1 sdt1p mdp1	
Esi0410_0009	bestclassified ambiguous ambiguous	sdt1p bcbf cof pset p5n1 tpp pmm psp s38k pmm mdp1 p5n1	
Esi0031_0026	unclassified bestclassified bestclassified	 spsc spsc	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0005_0257	unclassified unclassified unclassified		
Esi0094_0087	bestclassified ambiguous classified	nagd nagd hera bpgm	
Esi0125_0069	ambiguous unclassified unclassified	atpase hera	 1_53 2_37
Esi0129_0038	bestclassified unclassified classified	pmm bpgm	
Esi0367_0004	unclassified bestclassified bestclassified	 spsc spsc	
Esi0002_0208	unclassified unclassified unclassified		
Esi0100_0020	classified ambiguous unclassified	bpgm bpgm cof	
Esi0030_0056	unclassified bestclassified bestclassified	 spsc spsc	
Esi0968_0001	bestclassified bestclassified classified	tpp spsc tpp	
Esi0113_0015	bestclassified bestclassified bestclassified	ctd spsc spsc	
Esi0081_0006	classified bestclassified bestclassified	hera spsc spsc	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0315_0030	bestclassified classified classified	pmm pmm pmm	
Esi0014_0010	ambiguous bestclassified bestclassified	hera atpase psp mpgp cof spsc spsc	
Esi0110_0034	bestclassified bestclassified bestclassified	ctd ctd ctd	
Esi0268_0035	ambiguous ambiguous classified	hera atpase psp mpgp cof mpgp cof mpgp	
Esi0154_0054	bestclassified bestclassified unclassified	tpp tpp	
Esi0047_0015	bestclassified bestclassified bestclassified	mdp1 mdp1 mdp1	
Esi0012_0032	unclassified unclassified unclassified		
Esi0047_0025	classified bestclassified bestclassified	lpin spsc spsc	
Esi0016_0197	ambiguous ambiguous bestclassified	atpase hera atpase hera atpase	
Esi0033_0021	classified classified classified	enolase enolase enolase	
Esi0199_0012	classified unclassified unclassified	acidphosphatase	

Continue sur la page suivante

Tab. 1 – suite de la page précédente

Séquence	Statut	Familles	Nouvelles familles
Esi0163_0030	bestclassified unclassified unclassified	ctd	
Esi0080_0016	classified ambiguous unclassified	bpgm bpgm cof	
Esi0010_0223	ambiguous classified classified	nagd yhr100c hera yhr100c yhr100c	

t=1	t=3	t=5
0_3 Esi0048_0016 Esi0039_0126 Esi0243_0024 Esi0154_0054	**1_53** Esi0009_0142 Esi0023_0054 Esi0125_0069 Esi0009_0037	**2_37** Esi0009_0037 Esi0023_0054 Esi0009_0142 Esi0125_0069
	1_31 Esi0154_0054 Esi0004_0179 Esi0243_0024 Esi0064_0072 Esi0039_0126	

Tab. 2 – Nouvelles familles découvertes pour l'expérience de classification des séquence de *Ectocarpus*. Chaque colonne correspond aux familles découvertes pour chaque expérience faisant varier la valeur t (respectivement 1,3 et 5). Le nom de chaque famille est indiqué en gras.

C. Grammaire obtenue la superfamille des GH16

start

N0 → N0f103X | N0f151X | N0f39X | N0f73X | N0f83X | N0fnew1X

règles start des différentes familles

N0f103X → GXf103X1 Xf103X70 | GXf103X5 Xf103X70 | Gap GXf103X1 Xf103X70 | Gap Xf103X32 Plma15 Plma16 Xf103X35 | Gap Xf103X32 Plma15 Plma16 Xf103X4 | Gap Xf103X6 Plma225 Plma26 Plma27 Plma28 Xf103X25 | Gap Xf103X8 Plma21 Plma22 Plma23 GPlma24 Xf103X20 | Plma21 Plma22 Plma23 GPlma24 Xf103X20 | Plma5 Plma6 Plma7 GPlma8 Xf103X12 | Plma6 Plma7 GPlma8 Xf103X12 | Xf103X32 Plma15 Plma16 Xf103X35 | Xf103X32 Plma15 Plma16 Xf103X4 | Xf103X50 Plma5 Plma6 Plma7 GPlma8 Xf103X12 | Xf103X6 Plma225 Plma26 Plma27 Plma28 Xf103X25 | Xf103X8 Plma21 Plma22 Plma23 GPlma24 Xf103X20

N0f151X → Plma103 Plma104 Plma233 Plma106 Plma73 Plma46 Plma21 Plma22 Plma234 | Plma232 Plma103 Plma104 Plma233 Plma106 Plma73 Plma46 Plma21 Plma22 Plma234 | PlmaEx1 | Xf151X9 Plma232 Plma103 Plma104 Plma233 Plma106 Plma73 Plma46 Plma21 Plma22 Plma234

N0f39X → GPlma50 Xf39X5 Plma21 Plma22 Xf39X48 | Xf39X1 Plma28 GPlma29 Xf39X28 | Xf39X2 Xf39X5 Plma21 Plma22 Xf39X48 | Xf39X25 Plma79 Plma80 GPlma81 Xf39X97 | Xf39X5 Plma21 Plma22 Xf39X48

N0f73X → Xf73X11 Plma21 Plma22 Xf73X46

N0f83X → Xf83X10 Plma46 Plma21 Plma22 Plma23 Plma188 Xf83X11 Plma34 Plma35 Plma36 Xf83X6

N0fnew1X → GPlma50 Plma39 Plma40 Plma41 Plma42 Plma9 Plma10 Plma52 PlmaEx48 Plma56 PlmaEx49 Plma12 Plma13 Plma62 PlmaEx51 Plma15 Plma16 Plma17 Plma18 Plma19 PlmaEx32 Plma21 Plma22 Plma23 GPlma121 Plma69 Plma28 Plma29 Plma30 Plma31 Plma32 Plma49 | GXfnew1X26 GXfnew1X8 Xfnew1X29 Plma143 Plma144 Xfnew1X32 | GXfnew1X26 Plma151 Plma152 Plma130 Xfnew1X6 | GXfnew1X26 Xfnew1X8 Plma154 Plma155 Plma156 Plma157 Plma158 Plma159 Plma134 Plma135 Plma160 Xfnew1X52 Plma138 Plma139 Xfnew1X107 | GXfnew1X58 Xfnew1X31 | GXfnew1X8 Xfnew1X29 Plma143 Plma144 Xfnew1X32 | Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Plma160 Xfnew1X52 Plma138 Plma139 Xfnew1X107 | Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Xfnew1X65 | Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Plma160 Xfnew1X52 Plma138 Plma139 Xfnew1X107 | Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Xfnew1X65 | Plma151 Plma152 Plma130 Xfnew1X6 | Plma152 Plma130 Xfnew1X6 | Plma39 Plma40 Plma41 Plma42 Plma9 Plma10 Plma52 PlmaEx48 Plma56 PlmaEx49 Plma12 Plma13 Plma62 PlmaEx51 Plma15 Plma16 Plma17 Plma18 Plma19 PlmaEx32 Plma21 Plma22 Plma23 GPlma121 Plma69 Plma28 Plma29 Plma30 Plma31 Plma32 Plma49 | Plma4 Plma5 Plma6 GPlma7 Plma171 Plma143 PlmaEx83 | Plma40 Plma41 Plma42 Plma9 Plma10 Plma52 PlmaEx48 Plma56 PlmaEx49 Plma12 Plma13 Plma62 PlmaEx51 Plma15 Plma16 Plma17 Plma18 Plma19 PlmaEx32 Plma21 Plma22 Plma23 GPlma121 Plma69 Plma28 Plma29 Plma30 Plma31 Plma32 Plma49 | Plma5 Plma6 GPlma7 Plma171 Plma143 PlmaEx83 | PlmaEx3 Plma4 Plma5 Plma6 GPlma7 Plma171 Plma143 PlmaEx83 | Xfnew1X12 Plma133 Plma134 Plma135 Xfnew1X65 | Xfnew1X19 Plma137 Plma138 Plma139 Xfnew1X107 | Xfnew1X51 Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Plma160 Xfnew1X52 Plma138 Plma139 Xfnew1X107 | Xfnew1X51 Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Xfnew1X65

règles de subsubstitutabilité

Xf103X1 → GPlma209 GPlma210 GPlma211 GPlma212 GPlma213 GPlma214 GPlma215 GPlma216 GPlma217 GPlma218 GPlma219 GPlma220 GPlma221 Xf103X100 Plma13 Plma222 | GPlma210 GPlma211 GPlma212 GPlma213 GPlma214 GPlma215 GPlma216 GPlma217 GPlma218 GPlma219 GPlma220 GPlma221 Xf103X100 Plma13 Plma222 | GPlma211 GPlma212 GPlma213 GPlma214 GPlma215 GPlma216 GPlma217 GPlma218 GPlma219 GPlma220 GPlma221 Xf103X100 Plma13 Plma222 | Plma5 Plma6 Plma7 GPlma8 Plma42 Plma9 PlmaEx23 Plma55 Plma56 Plma57 PlmaEx50 Plma59 Xf103X5 | Plma6 Plma7 GPlma8 Plma42 Plma9 PlmaEx23 Plma55 Plma56 Plma57 PlmaEx50 Plma59 Xf103X5 | Xf103X50 Plma5 Plma6 Plma7 GPlma8 Plma42 Plma9 PlmaEx23 Plma55 Plma56 Plma57 PlmaEx50 Plma59 Xf103X5

Xf103X100 → Plma61 | PlmaEx105

Xf103X109 → GPlma66 Plma88 Plma21 Plma22 GPlma47 Plma69 Plma28 PlmaEx41 | PlmaEx106 Plma22 Plma23 GPlma107 Xf103X20

Xf103X12 → Plma42 Plma9 PlmaEx23 Plma55 Plma56 Plma57 PlmaEx50 Plma59 N0f103X | PlmaEx22 Plma72 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma62 PlmaEx52 Xf103X70

Xf103X20 → Plma224 Plma225 Plma26 Plma27 Plma28 Xf103X25 | PlmaEx40 Plma33 Plma34 Plma35 Plma36 Plma37 PlmaEx46

Xf103X25 → GPlma226 Plma227 | Plma29 Plma30

Xf103X32 → GXf103X1 Plma223 | Xf103X5 Plma230

Xf103X33 → Plma85 Plma228 PlmaEx106 Plma22 Plma23 Plma107 | PlmaEx29 Plma21 Plma22 Plma23 Plma24

Xf103X35 → GPlma64 Plma78 Plma79 Plma80 GPlma81 Plma208 Plma46 Plma21 Plma22 PlmaEx34 | GXf103X33 Xf103X20 | Plma85 Plma228 Xf103X109

Xf103X4 → Plma17 GPlma18 Xf103X86 | Plma85 Plma228 Xf103X109

Xf103X5 → Plma13 Plma62 Plma229 | Plma42 Plma9 Plma10 Plma11 PlmaEx26 Plma12 Plma13 Plma14 Plma125 | Plma5 Plma6 Plma7 Plma8 PlmaEx22 Plma72 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma62 PlmaEx52 | Plma6 Plma7 Plma8 PlmaEx22 Plma72 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma62 PlmaEx52 | Plma60 Xf103X100 Plma13 Plma62 Plma229 | Plma9 Plma10 Plma11 PlmaEx26 Plma12 Plma13 Plma14 Plma125 | PlmaEx9 Plma42 Plma9 Plma10 Plma11 PlmaEx26 Plma12 Plma13 Plma14 Plma125 | Xf103X100 Plma13 Plma62 Plma229 | Xf103X50 Plma5 Plma6 Plma7 Plma8 PlmaEx22 Plma72 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma62 PlmaEx52

Xf103X50 → Plma4 | PlmaEx6

Xf103X6 → Xf103X32 Plma15 Plma16 GXf103X33 Plma224 | Xf103X32 Plma15 Plma16 Plma17 GPlma18 Plma126 PlmaEx68 Plma120 Plma21 Plma22 PlmaEx37 | Xf103X8 Plma21 Plma22 Plma23 GPlma24 Plma224

Xf103X70 → Plma223 Plma15 Plma16 Xf103X35 | Plma230 Plma15 Plma16 Xf103X4

Xf103X8 → Plma86 | PlmaEx12 Plma86 | Xf103X32 Plma15 Plma16 PlmaEx29

Xf103X86 → Plma126 PlmaEx68 Plma120 Plma21 Plma22 PlmaEx37 Plma225 Plma26 Plma27 Plma28 Xf103X25 | Plma19 PlmaEx31

Xf151X9 → Plma231 | PlmaEx18

Xf39X1 → GPlma50 Xf39X5 Plma21 Plma22 Xf39X17 | GXf39X8 GXf39X29 Plma26 Plma27 | Xf39X2 Xf39X5 Plma21 Plma22 Xf39X17 | Xf39X37 Plma13 Plma14 Xf39X96 Plma16 Plma17 GXf39X4 Plma26 Plma27 | Xf39X5 Plma21 Plma22 Plma23 GPlma24 Plma25 Plma26 Plma27 | Xf39X5 Plma21 Plma22 Xf39X17 | Xf39X69 Plma9 Plma10 Plma11 Xf39X3

Xf39X111 → PlmaEx10 | PlmaEx11

Xf39X13 → GPlma74 GPlma75 Plma76 | GPlma75 Plma76 | Xf39X111 GPlma74 GPlma75 Plma76 | Xf39X69 Plma9 Plma10 Plma11 PlmaEx27 Plma43 Plma44

Xf39X133 → Plma11 Plma72 | Plma52

Xf39X16 → PlmaEx25 Plma12 | PlmaEx27 Plma43 GPlma44 Plma45

Xf39X17 → Plma47 GPlma48 Plma26 Plma27 | Plma68 Plma69

Xf39X19 → GPlma32 Xf39X99 | Plma70 Plma35 Plma71

Xf39X2 → GPlma38 Plma39 Plma40 | GPlma50 Plma51 | Plma39 Plma40 | Plma40 | Plma51

Xf39X24 → GXf39X76 Plma4 Plma5 Plma6 Plma7 | Plma4 Plma5 Plma6 Plma7 | Plma5 Plma6 Plma7 | Plma5 Plma6 PlmaEx19 | Plma6 PlmaEx19 | PlmaEx4 Plma5 Plma6 PlmaEx19

Xf39X25 → GXf39X24 GPlma83 GPlma84 Xf39X96 Plma16 PlmaEx30 | Xf39X37 Plma13 Plma14 Xf39X96 Plma16 Plma17 GPlma77 Plma78

Xf39X28 → Plma30 Plma31 Xf39X19 | Plma82

Xf39X29 → Plma46 Plma21 Plma22 Plma47 Plma48 | PlmaEx33 Plma21 Plma22 Plma23 GPlma24 Plma25

Xf39X3 → GXf39X67 GXf39X29 Plma26 Plma27 | GXf39X67 Plma46 Plma21 Plma22 Xf39X17 | Plma72 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 Plma57 Plma58 Plma59 Plma60 Plma61 Plma13 Plma62 GPlma63 Xf39X96 Plma16 Plma64 Plma65 Plma66 GPlma67 Xf39X38 Plma21 Plma22 Xf39X17 | Xf39X16 Plma13 Plma14 Xf39X96 Plma16 Plma17 GXf39X4 Plma26 Plma27

Xf39X32 → Plma18 Plma19 Plma20 PlmaEx33 Plma21 Plma22 Plma23 Plma24 | Plma77 PlmaEx55 Plma80 PlmaEx56 Plma21 Plma22 PlmaEx35

Xf39X37 → GXf39X13 Plma45 | Xf39X69 Plma9 Plma10 Plma11 Xf39X16

Xf39X38 → Plma73 Plma46 | PlmaEx54

Xf39X4 → GPlma77 Xf39X65 | GXf39X32 Plma25 | Plma18 Plma19 GPlma20 Xf39X29

Xf39X40 → GPlma83 GPlma84 Xf39X96 Plma16 Xf39X83 | Plma8 PlmaEx21 Plma9 Plma10 Plma11 Xf39X67 PlmaEx33

Xf39X48 → Plma23 Xf39X56 | Xf39X17 Plma28 GPlma29 Xf39X28

Xf39X5 → GXf39X24 Xf39X40 | GXf39X50 Xf39X38 | GXf39X76 GXf39X24 Xf39X40 | GXf39X8 Plma46 | Xf39X25 Plma79 Plma80 GPlma81 Plma88 | Xf39X8 PlmaEx33

Xf39X50 → Plma89 Plma90 Plma91 Plma92 Plma93 Plma94 Plma95 Plma96 Plma97 Plma98 Plma99 Plma12 Plma100 Plma101 Plma102 Plma103 Plma104 Plma105 Plma106 | Plma90 Plma91 Plma92 Plma93 Plma94 Plma95 Plma96 Plma97 Plma98 Plma99 Plma12 Plma100 Plma101 Plma102 Plma103 Plma104 Plma105 Plma106 | Plma91 Plma92 Plma93 Plma94 Plma95 Plma96 Plma97 Plma98 Plma99 Plma12 Plma100 Plma101 Plma102 Plma103 Plma104 Plma105 Plma106 | Xf39X69 Plma9 Plma10 GXf39X133 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 Plma57 Plma58 Plma59 Plma60 Plma61 Plma13 Plma62 GPlma63 Xf39X96 Plma16 Plma64 Plma65 Plma66 Plma67

Xf39X56 → GPlma24 Plma25 Plma26 Plma27 Plma28 GPlma29 Xf39X28 | GXf39X78 Xf39X99 | Plma107 Plma108 Plma109 PlmaEx63 | PlmaEx39 Plma35 Plma36 Plma87

Xf39X65 → Plma78 Plma79 Plma80 Plma81 PlmaEx57 Plma21 Plma22 PlmaEx36 | PlmaEx55 Plma80 PlmaEx56 Plma21 Plma22 PlmaEx35 Plma25

Xf39X67 → Plma72 Plma53 Plma43 Plma44 Plma54 Plma55 Plma56 Plma57 Plma58 Plma59 Plma60 Plma61 Plma13 Plma62 GPlma63 Xf39X96 Plma16 Plma64 Plma65 Plma66 GPlma67 Plma73 | Xf39X16 Plma13 Plma14 Xf39X96 Plma16 Plma17 Plma18 Plma19 Plma20

Xf39X69 → GPlma38 Plma39 Plma40 Plma41 Plma42 | GXf39X76 Xf39X24 Plma8 PlmaEx21 | Plma39 Plma40 Plma41 Plma42 | Plma40 Plma41 Plma42 | Plma41 Plma42 | Plma42 | Plma51 Plma41 Plma42 | Xf39X24 Plma8 PlmaEx21

Xf39X76 → Plma3 | PlmaEx2

Xf39X78 → GPlma24 Plma25 Plma26 Plma27 Plma28 Plma29 Plma30 Plma31 Plma32 | PlmaEx38

Xf39X8 → GXf39X50 Plma73 | Xf39X37 Plma13 Plma14 Xf39X96 Plma16 Plma17 Plma18 Plma19 Plma20 | Xf39X69 Plma9 Plma10 Plma11 Xf39X67

Xf39X83 → GPlma85 Plma86 | PlmaEx30 Plma79 Plma80 GPlma81 Plma88

Xf39X96 → Plma15 | PlmaEx53

Xf39X97 → Plma88 Plma21 Plma22 Xf39X48 | PlmaEx57 Plma21 Plma22 PlmaEx36 Plma26 Plma27 Plma28 GPlma29 Xf39X28

Xf39X98 → PlmaEx44 | PlmaEx45

Xf39X99 → Plma33 Plma34 Plma35 Plma36 Plma37 Xf39X98 | Plma49

Xf73X1 → GPlma3 Plma93 PlmaEx58 | GXf73X15 Plma97 | Plma92 Plma93 Xf73X27 | Plma93 PlmaEx58 | PlmaEx58 | Xf73X16 Plma92 Plma93 Xf73X27 | Xf73X3 Xf73X16 Plma92 Plma93 Xf73X27

Xf73X11 → GXf73X18 Plma119 Plma120 | Xf73X13 Plma101 Plma102 Plma103 Plma104 Plma105 Plma106 Plma73 Plma46

Xf73X13 → Plma92 Plma93 Plma94 Xf73X19 | Xf73X1 Plma98 Plma99 Plma12 Plma100 | Xf73X16 Plma92 Plma93 Plma94 Xf73X19

Xf73X15 → Plma92 Plma93 PlmaEx59 Plma115 | Xf73X16 Plma92 Plma93 Plma94 Plma95 Plma96 | Xf73X16 Plma92 Plma93 PlmaEx59 Plma115 | Xf73X3 Xf73X16 Plma92 Plma93 PlmaEx59 Plma115

Xf73X16 → Plma114 | Plma90 Plma91 | Plma91 | PlmaEx13 Plma91 | Xf73X28 Plma91 | Xf73X3 Plma114

Xf73X18 → Plma12 PlmaEx28 Plma102 Plma103 | Plma40 Plma41 Plma42 Plma9 Plma10 PlmaEx24 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma14 Plma125 PlmaEx67 Plma15 Plma16 Plma17 GPlma18 Plma126 | Plma41 Plma42 Plma9 Plma10 PlmaEx24 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma14 Plma125 PlmaEx67 Plma15 Plma16 Plma17 GPlma18 Plma126 | PlmaEx28 Plma102 Plma103 | PlmaEx7 Plma12 PlmaEx28 Plma102 Plma103 | PlmaEx8 Plma40 Plma41 Plma42 Plma9 Plma10 PlmaEx24 Plma44 Plma54 Plma55 Plma56 GPlma124 Plma12 Plma13 Plma14 Plma125 PlmaEx67 Plma15 Plma16 Plma17 GPlma18 Plma126

Xf73X19 → Plma95 Xf73X43 Plma98 Plma99 Plma12 Plma100 | PlmaEx60 Plma112 Plma98 PlmaEx61

Xf73X20 → Plma107 Plma108 Xf73X7 | Plma121 PlmaEx66

Xf73X27 → Plma94 Plma95 Xf73X43 | PlmaEx59 GPlma115 Plma97

Xf73X28 → GPlma122 Plma90 | GXf73X49 Plma90 | Plma90 | PlmaEx13

Xf73X3 → Plma113 | PlmaEx14

Xf73X32 → Plma110 Xf73X57 | Plma117 Xf73X41 | PlmaEx63 | PlmaEx64

Xf73X41 → Plma118 | PlmaEx65

Xf73X43 → Plma115 Plma116 Plma112 | Plma96 Plma97

Xf73X46 → Plma23 Xf73X20 | Plma47 Plma48 PlmaEx47 Plma27 Plma28 Plma29 PlmaEx42 Plma33 Plma34 Plma35 Plma36 PlmaEx43

Xf73X49 → Plma122 | Plma89

Xf73X57 → Plma111 | Plma123

Xf73X7 → Plma109 Xf73X32 | PlmaEx62

Xf83X10 → Xf83X16 Xf83X66 | Xf83X4 Xf83X35

Xf83X11 → Plma189 Xf83X57 | Xf83X7 Plma33

Xf83X13 → Plma175 Plma74 Plma176 Plma115 Plma177 | Plma176 Plma115 Plma177 | Plma195 Plma196 Plma6 Plma197 Plma198 GPlma199 Plma200 | Plma196 Plma6 Plma197 Plma198 GPlma199 Plma200 | Plma6 Plma197 Plma198 GPlma199 Plma200 | Plma74 Plma176 Plma115 Plma177

Xf83X16 → Plma5 Plma6 Plma7 PlmaEx20 Plma178 PlmaEx96 Plma183 PlmaEx98 | Plma6 Plma7 PlmaEx20 Plma178 PlmaEx96 Plma183 PlmaEx98 | PlmaEx5 Plma5 Plma6 Plma7 PlmaEx20 Plma178 PlmaEx96 Plma183 PlmaEx98 | Xf83X4 Plma181 Plma182 Plma183 Plma184 PlmaEx99 | Xf83X4 Plma182 Plma183 Plma201 Plma186 Plma202

Xf83X21 → Plma185 Plma186 Plma187 Plma73 | PlmaEx99 Xf83X66

Xf83X30 → Plma190 Plma191 | Plma203 Plma204

Xf83X31 → Plma205 | PlmaEx101

Xf83X35 → Plma181 Plma182 Plma183 Plma184 Xf83X21 | Plma182 Plma183 Plma201 Plma186 Plma202 Xf83X66

Xf83X4 → Plma196 Plma6 GPlma7 Plma198 PlmaEx102 Plma178 Plma179 PlmaEx97 | Plma6 GPlma7 Plma198 PlmaEx102 Plma178 Plma179 PlmaEx97 | PlmaEx17 Plma196 Plma6 GPlma7 Plma198 PlmaEx102 Plma178 Plma179 PlmaEx97 | Xf83X13 Plma178 Plma179 Plma180

Xf83X44 → GPlma193 Plma194 | GPlma206 Plma207

Xf83X57 → Plma203 Xf83X63 | Xf83X30 Plma33

Xf83X58 → Plma189 Plma190 | PlmaEx100

Xf83X6 → Plma37 Plma192 GXf83X31 Xf83X44 | Plma87

Xf83X63 → Plma204 Plma33 | PlmaEx104

Xf83X66 → Plma208 | PlmaEx103

Xf83X7 → Plma189 Xf83X30 | Xf83X58 Plma191

Xfnew1X107 → Plma163 Plma164 Xfnew1X39 Plma167 Plma140 GXfnew1X47 Xfnew1X31 | PlmaEx79 Plma140 GXfnew1X47 Xfnew1X31

Xfnew1X119 → PlmaEx75 Plma136 PlmaEx78 Plma137 Plma138 Plma139 PlmaEx79 | PlmaEx77 Plma139 Plma163 PlmaEx90 Plma166 Plma167

Xfnew1X12 → GXfnew1X26 Plma151 Plma152 Plma130 Xfnew1X59 | GXfnew1X8 Plma132 | Plma146 Plma147 Plma128 GPlma129 Plma152 Plma130 PlmaEx73 Plma156 Plma157 | Plma147 Plma128 GPlma129 Plma152 Plma130 PlmaEx73 Plma156 Plma157 | Plma151 Plma152 Plma130 Xfnew1X59 | Plma152 Plma130 Xfnew1X59 | Xfnew1X51 Plma146 Plma147 Plma128 GPlma129 Plma152 Plma130 PlmaEx73 Plma156 Plma157

Xfnew1X17 → Plma173 | PlmaEx94

Xfnew1X19 → GXfnew1X26 Xfnew1X8 Plma154 Plma155 Plma156 Plma157 Plma158 Plma159 Plma134 Plma135 Plma160

Plma161 Plma136 Plma162 | Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Plma160 Plma161 Plma136 Plma162 | Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Plma160 Plma161 Plma136 Plma162 | Xfnew1X12 Plma133 Plma134 Plma135 PlmaEx75 Plma136 PlmaEx78 | Xfnew1X51 Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Plma160 Plma161 Plma136 Plma162

Xfnew1X21 → Plma160 Xfnew1X52 Plma138 Plma139 Plma163 Plma164 Xfnew1X39 Plma167 Plma140 Xfnew1X47 | Xfnew1X119 Plma140 Xfnew1X47

Xfnew1X26 → Plma38 | Xfnew1X34 Plma150

Xfnew1X29 → Plma132 Plma133 Plma134 Plma135 PlmaEx76 Plma139 PlmaEx80 Plma140 Plma141 PlmaEx81 | Plma154 Plma155 Plma156 Plma157 Plma158 Plma159 Plma134 Plma135 GXfnew1X21 Plma171

Xfnew1X31 → Plma171 Plma143 Plma144 Xfnew1X32 | PlmaEx82 Plma143 Plma144 PlmaEx84

Xfnew1X32 → Plma172 Xfnew1X17 | Plma174

Xfnew1X34 → Plma128 PlmaEx69 | Plma146 Plma147 Plma128 Plma148 Plma149 | Plma147 Plma128 Plma148 Plma149 | Plma147 Plma128 PlmaEx69 | PlmaEx16 Plma147 Plma128 PlmaEx69 | Xfnew1X51 Plma146 Plma147 Plma128 Plma148 Plma149

Xfnew1X38 → Plma170 | PlmaEx93

Xfnew1X39 → Plma165 Plma166 | PlmaEx91

Xfnew1X41 → Plma131 Plma153 Plma154 Plma155 Plma156 Plma157 Plma158 Plma159 Plma134 Plma135 Plma160 Xfnew1X52 Plma138 Plma139 Plma163 Plma164 Plma165 | Xfnew1X59 Plma133 Plma134 Plma135 PlmaEx77 Plma139 Plma163 PlmaEx90

Xfnew1X43 → GPlma129 Plma152 Plma130 PlmaEx73 Plma156 Plma157 Plma133 | Plma148 Plma149 Plma150 Xfnew1X8 Plma154 Plma155 Plma156 Plma157 Plma158 Plma159

Xfnew1X47 → Plma141 Plma142 | Plma168 Plma169 Xfnew1X38

Xfnew1X51 → Plma145 | PlmaEx15

Xfnew1X52 → Plma161 Plma136 Plma162 Plma137 | PlmaEx88

Xfnew1X58 → GXfnew1X26 Plma151 Plma152 Plma130 Xfnew1X41 Plma166 Plma167 Plma140 Xfnew1X47 | GXfnew1X26 Xfnew1X8 Plma154 Plma155 Plma156 Plma157 Plma158 Plma159 Plma134 Plma135 Xfnew1X21 | Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Xfnew1X21 | Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Xfnew1X21 | Xfnew1X12 Plma133 Plma134 Plma135 Xfnew1X21 | Xfnew1X19 Plma137 Plma138 Plma139 Plma163 Plma164 Xfnew1X39 Plma167 Plma140 Xfnew1X47 | Xfnew1X51 Plma146 Plma147 Plma128 Xfnew1X43 Plma134 Plma135 Xfnew1X21

Xfnew1X59 → Plma131 GPlma153 Plma132 | PlmaEx72 Plma156 PlmaEx86

Xfnew1X6 → Plma131 GPlma153 Xfnew1X29 Plma143 Plma144 Xfnew1X32 | Plma131 Plma153 Plma154 Plma155 Plma156 Plma157 Plma158 Plma159 Plma134 Plma135 Plma160 Xfnew1X52 Plma138 Plma139 Xfnew1X107 | Xfnew1X41 Plma166 Plma167 Plma140 GXfnew1X47 Xfnew1X31 | Xfnew1X59 Plma133 Plma134 Plma135 Xfnew1X65

Xfnew1X65 → GXfnew1X21 Xfnew1X31 | PlmaEx75 Plma136 PlmaEx78 Plma137 Plma138 Plma139 Xfnew1X107 | PlmaEx76 Plma139 PlmaEx80 Plma140 Plma141 PlmaEx81 Plma143 Plma144 Xfnew1X32

Xfnew1X8 → GXfnew1X26 Plma151 Plma152 Plma130 Plma131 Plma153 | Plma127 Plma128 Plma129 PlmaEx70 Plma130 Plma131 PlmaEx74 | Plma128 Plma129 PlmaEx70 Plma130 Plma131 PlmaEx74 | Plma129 PlmaEx70 Plma130 Plma131 PlmaEx74 | Plma151 Plma152 Plma130 Plma131 Plma153 | Plma152 Plma130 Plma131 Plma153

règles définissant les blocs plma

Plma10 → ATc
Plma100 → IVc NV NS NS NF
Plma101 → FYc NT GYc ETc GWc DPc ASTWc
Plma102 → DEHc DGNc
Plma103 → DNRTc KPQVc RWc DVc NE ILVc ND FIc NE FIVc NL NG
Plma104 → KNYc DNRc STc GTc KQRc
Plma105 → AIVc NQ NF NN WYc FYc KTc
Plma106 → HNc EGc AKQVc NG DGHKNc HKNRc NE FHKQYc FILRVYc AHIVYc DNSYc
Plma107 → EKQRTVc AEKLc KVc FHYc

Plma108 → RTc AGc TYc ADENQRTc DNQc NI NP ADKQSTVc ATc NP GQc NK NI NM NM NN ALc NW NPc NG AIKLTVc GTc NV ND DEGSc NW LTc GKRc APRSVc FYc DNc NG
Plma109 → ARVc NTc NPc ILc QSTYc ARc EHSYc LYc DEHQc WYc MVc KRSTc NY
Plma11 → DNc
Plma110 → NT
Plma111 → NK KRc
Plma112 → ASc RYc NPc
Plma113 → NM NK NN NR NV NI NS NL NL NM NA NS NL NL NL NV NL NS NV NI NV NA NP NF NY NK NA NE NA NA NT NV NV NN NT NP NF NV NA NV NF NS NN NF
Plma114 → DEc NS NSc QRc FWc
Plma115 → NN NG NK LMc IKc NL NT ILTc DKc NR DEc
Plma116 → NY NG NG
Plma117 → NYc KPc NN NG
Plma118 → NV NP NQ ND NN NP NT NP NT NP NT NI NA NP NS NT NP NT NN NP NN NL NP NL NK NG ND NV NN NG ND NG NH NV NN NS NS ND NY NS NL NF NK NR NY NL NL NR NV NI ND NR NF NP NV NG ND NQ NS NV NA ND NV NN NR ND NG NR NI ND NS NT ND NL NT NM NL NK NR NY NL NI NR NA NI NP NS NL
Plma119 → NP
Plma12 → NY NG HIKLMRYc FIVYc DEQc ACFIVc NRSc AIMc KQRWc ALMPVc AIMPc AKNSTc AGNc DKPQRSTVc NG
Plma120 → AGTc NA DERc DQTc
Plma121 → NQ NE
Plma122 → NM NK NK NK NS CWc NF NT NL MVc ITc NT FGc AVc FIc NS NL FIc NF
Plma123 → NS NN
Plma124 → AVc QSc
Plma125 → NG
Plma126 → GSc STc LVc NH NT EKc AKc FYc NN NH LVc LMc NG NT NQ
Plma127 → HQc NA AQc ADc ALc DEc NW DEc
Plma128 → DEQSc FLYc PSc IVc NP NA ADQc APc GPc AEGMc GNc KMQRYc ERSTc NW EQc NL
Plma129 → LVc
Plma13 → LNTVc FWc NP NA FILc NW LMc LMc
Plma130 → GKNRc NW DLNRTVc ADEc AGHQSYc FHKYc IKMPc DHNc AGNPSc NW ILPQSc NG NP AGc ADFLNRc GSTc DEFLNVc FWYc EKRSc ASTc DEGNRc HNc NS
Plma131 → FWYc
Plma132 → NY IVc NT NS KRc KTc PQc LVc ERc PYc NP NL NY NT NE NV LRc NM
Plma133 → KQRc AVc NS GPSc LTc KPVc
Plma134 → LMc
Plma135 → ASc NSTc AGNc FVc NW LMc NL NS ADERYc ND DNSc ETVc ENQRTc NE ILc ND AIVc AIMVc NE ACISc NY NG DGNSTc
Plma136 → KQc HRc NM HNc LTVc ASc HYc NH IVc NF IQVc NR DENc NP NF QTc DEc
Plma137 → GNc GQc EPc LTc LYc NR NN GKc NF HRc
Plma138 → NR IYc
Plma139 → AGc AVc FHYc NW IKQRc DNSc ADPc FRTWYc EHNYc CFILVc DEc FIYc FYc FILVYc DNc NG
Plma14 → NP CDKQRc DNRWc NQSc DEHQTVc FYc
Plma140 → DNSc EKQRSc PSc NM HRc ILVc ILc FLc DNTc LMTVc NE ADEHSYc HQc DPQTc NW NR
Plma141 → KVc DKQc ARSc NG IVc
Plma142 → DEc
Plma143 → DENRc NL ALc ND AEPSWc GNSTc IKLQRc NSc IKNRTc AFILMTc QRWYc IVYc DRc NW ILVc NR
Plma144 → TVc LYc KQRc
Plma145 → FIc NM ASc NI ITc NS AVc NQ AGc NA
Plma146 → DEc NW
Plma147 → ASc
Plma148 → DQc PSc NV NS ND DEc NF NN NY
Plma149 → NI NA PQc PTc NN NN
Plma15 → AGNSTVc DGNPRSTYc
Plma150 → NK APTc ASc AEc FLc
Plma151 → GNTc
Plma152 → DENRSTc
Plma153 → LTc TVc NG NG
Plma154 → AVc NL NG NL NQ NA NT NE NK ATc
Plma155 → NG NT NSc EKc
Plma156 → LVc KLNYc ACLSc NG AIVc IVc NS STc HKTc AEc ATc FLc STc NY NP LVc FYc LMVc NE
Plma157 → AVc MRSc IVc
Plma158 → NK
Plma159 → NP NT NN NN NT
Plma16 → NW APc ADHIKRTWYc CDNQSYc GVc NE ILMTVc ND IVc FMTVWc NE
Plma160 → DSc KRc
Plma161 → IPVc NG NQ NE NW FYc ADc
Plma162 → NY NQ NP KTc ND AGSc NG STc NW IVYc HYc DNc
Plma163 → EVc HLRc NV NR STc
Plma164 → ITVc STc
Plma165 → NG EPc GNc NI NI ND NP EKQc
Plma166 → GNc DYc FTc DNc NG KTc
Plma167 → NG FLc
Plma168 → DSc
Plma169 → AVc NK
Plma17 → AEHSVYc RVc NG
Plma170 → NP NSc ASc
Plma171 → AEQRSc
Plma172 → ASc
Plma173 → NQ
Plma174 → NP
Plma175 → NA NSc MPc NQ NP HPc NI NA NK NP NG NE NT NW NI NL NQ AEc NK NR NS ND NE NF DNc NV NK ND NA
Plma176 → KRc NN NE NN NA NT NV NS
Plma177 → STc NH NQc NR NT NF NW ND NG NC NN NQ NQ NQ NV NA NN NY NP NL NY

Plma178 → FYc NT SVc AGc IMVc AFLc KQc STc KRYc AQc KTVc FGTc NTc NY NG NY FYc NE NA KRc NI KQc NG ACSc DNPSc IRTVc FGc EPc NG LVc CSc NP ASc NF NW LMc NY NS
Plma179 → DKTc FIc ND DRYc STc ILVc
Plma18 → FKQRVc DLNRTc NP
Plma180 → ATc
Plma181 → EKc
Plma182 → EKNc EGc ADEc
Plma183 → ITVc QRTVc NY NS NE NI ND IVc NV NE LMc LQSTc NQ
Plma184 → KRc
Plma185 → GSc ANc NV NR NE NS ND NH ND
Plma186 → NL NH ANc IVc NV KVc EKc NN NG KQSc GPc TVc NW KMc NR NP GKc MSc AFYc NP EQc
Plma187 → NT NN NH NN NG NY NH
Plma188 → EIVc EIKQc NV AGc ERYc KQSc DEKPc NN HKLTc FNYc NW NH LRc EPQc NM NTc LVc ATc ILc STc LMQc EGc NL NR AEKPc NP
Plma189 → FHYc
Plma19 → GNQc QRSc ILVc
Plma190 → NT EQc NW KRc NC NN NQ NF NY NP NS NA
Plma191 → NTc AKc
Plma192 → SVc AGc GNc NN
Plma193 → NS NC NP DNc NT FWc NV NA NV NTc GSc NV NQc LVc STc
Plma194 → NT NL RSc NK NG NQ NS TVc NT NL ENc STc NTc NV NL NP ANc NC NA NT NN NK KNc NV IVc NY NS STc NS NN NK NN NV NA NT NV NTc NSc AEc NG NV NV NK NA NK NN NK NG STc NA NT NI NT NV NK NT NK NN NK NG NK ITc ND KTc LVc MTc NI ATc NV NN
Plma195 → NM NK NK NP NN NF NY NG NK NM NG NR NT NA NL NS NS NL NF NY NL NF NF NL NG NL NV NY NG NQ NQ NP NT NK
Plma196 → NT ASc KNc NP GNc DEc KQc NW ATc NI NK NW NSc ARc
Plma197 → NI NK NT NG
Plma198 → NN LPc EPc NN FTc GSc NA NW KTc NW DNc NN EQc KTc NN AVc KVc AIc ASc NN NG IVc
Plma199 → NA NE NL NT NM NR NH NN NA NN NN NT
Plma20 → LSc NG NT ILc NH NF NG
Plma200 → NP ND NG NG NT NY
Plma201 → NF ND NW NY NE NG NH NQ ND ND NI NY ND NM ND NL NN
Plma202 → NE NQ NL NN NK NW NR
Plma203 → AVc KVc NF FGc DNc NN KRc
Plma204 → NN NN NA NI NN NP NE NT ND NA NK NA NR NE
Plma205 → NT
Plma206 → NT NN NP NP NT NS NE NV NG NT NL NK NTc NK NG
Plma207 → ILc DTc NH NW ND NA NS NT NG NT NI NS NA NV NS NN NN NT NK NT NG NQ NY NA NG NS NV NN NN NA NS NI NA NQ NI NV NT NL NK NA NN NT NS NY NK NV NS NA NF NG NK NA NS NS NP NG NT NS NA NY NL NG NI NS NK NA NS NN NN NE NL NI NS NN NF NE NF NK NT NT NS NY NS NK NG NE NI NE NI NR NT NG NN NV NQ NE NS NY NR NI NW NY NW NS NS NG NQ NA NY NC ND ND NF NN NL NV NE NI NN NS NG NA NS NQ NL NN NE NN NE NT NE NT NA NL NE NK NG NI NH NI NY NP NN NP NY NK NN NG NP NL NT NI ND NF NG NK NP NF NS NG NE NV NQ NI NT NG NL NN NG NR NT NF NL NR NR NN NV NV ND NQ NT NS NV NQ NL NL NE NS NK NS NK NF NK NS NG NL NY NI NV NK NI NS NG NP ND NG NE NV NS NK NK NI NL NV NE
Plma208 → ASc IPc
Plma209 → NV NL AGc AFc NQ NR NS NR FYc AGc NS NN NT NF NG NA
Plma21 → ADEGRSc
Plma210 → EKc NE GSc NR FVc NM NL NI NG NL NG NK NR NK DEc NR APc NG NQ NS
Plma211 → STc NK NV GTc NA NG KVc NW CFc ND NA NV NF NP NI KSc NG
Plma212 → NA NK NA NG EQc KTc ILc NT NP TVc NF QSc NY KTc NG NSc NW NM NN NG FYc NV NY NL ND MVc DKc NQ DNc NG NR NF DNc NV
Plma213 → NE STc AIc NSc KNc AGc NN NV NL NN NP NP NA NF NT ILc NP AKc ND NL KPc NPc NG NI NY NR NL NR NY NK NV ND NW NS ESc AVc DNc NP AGc NG NRc NT
Plma214 → KQc IVc NP NF NG KQc NP FYc NT IVc NK MVc KNc NP AEc NPc NG NF
Plma215 → NK DNc ND KTc FYc TVc NI NP NA NE NY MVc ND NG ND IVc NE NI
Plma216 → KSc NG NRc AGc LMc NH GYc NY NL NY IVc ND FLc NN NN ND NG VYc NF STc NP
Plma217 → NE NL NV ASc NY NS FHc NY DKc NG KYc NN NS NL NG NE ETc IVc
Plma218 → LMc NP ALc NG NVc NY NR NA NR LYc NK IVc ND WYc DSc NN NI ND NP NA NG NQ NW AEc EQc NG
Plma219 → NQc HQc NI ADc ASc NN NG NG NY IVc NV ND NF NL NL NN NV NH APc NE NK
Plma22 → FMWYc HNc DILNTVc FYc AGKQRSTc CFILMVc DEFILMNTVc NVWc ADEGNQRSTc ADEHKPQRVc DEGHKNSTc DEHKLNQSTYc ILMVc ADHIKQRSTVc FMTVWc FLSTYc FILVYc DNc
Plma220 → NA ETc NG NY NA NA DKc ESc NL NI IVc KRc NH NG NH DNc NL DNc NG DPc NQ NY IVc HMc NG NN NR NQ NW KSc NE
Plma221 → AGc NI HQc NS EKc NK KTc NF AGc NF NT NY NG NK NV
Plma222 → NP EVc ND GQc
Plma223 → KQRc
Plma224 → DNc AHc NL
Plma225 → DNSc
Plma226 → KNc NG DSc NW NA NA AHc APc ND
Plma227 → GNc FHc NT NY NE FTc FLc FIc ND NW NV NR NV NY NQ NK KNc NG
Plma228 → ASc
Plma229 → ND ND
Plma23 → NG
Plma230 → AGc
Plma231 → NM NT NM
Plma232 → ATc NL STc NSc NA NF LVc FLc ILc NF NS NC FVc NL NA CIc NS FIc CSc NV SWc NG KRc NP NA NT NF NL DEc ND NF QRc NI NT NW NS ND NS HRc NI KRc NQ NI ND NG NG KRc NA NI NQ NL IVc NL ND NQ NN NS NG NC NG NF NA NS NK NR KRc NY NL NF NG NR NV NS NM NK NI NK NL NI NP NG ND ASc NA NG NT NV NT NA NF NY NM NN NS ND NT ND
Plma233 → NP NY NT NV NQ NT NN NI NY NA
Plma234 → DEc NV NP FIc NR NV NY NK NN NN NE NA KRc NN NI NP FYc NP NK NF NQ NP NM NG NV NY NS NT NL NW NE NA ND ND NW NA NT NR NG NG NL NE NK NI ND NW NS NK NA NP NF NL NA SYc NY NK ND NF ND NI NE NG NC NP NV NP NG NP NA NSc NC NA NS NN PTc GRc NSc NW NW NE NG STc ATc NY NQ NA NL NN AVc NM NE NA NR NR NY NR NW NV NR IMc NN NH IMc NI NY ND NY NC NT ND NK NS NR NY NP NV ISTc NP NP NE

NC IMc NA NG NI
Plma24 → NK
Plma25 → GNc
Plma26 → NQSTVc EGKNPc AGMNQSc ASWc
Plma27 → NP
Plma28 → FYc DENQRc EHKQc DEHNPRc FHc FHYc FILVc ILVc ILMc NN ILQVc ASc CLVWc AGc
Plma29 → NG
Plma3 → KNc NG NQ NW NV ND NL NS NN NS NS
Plma30 → DNc
Plma31 → NF
Plma32 → DFGc GQc NG
Plma33 → DKNSTc AELPc
Plma34 → ADESTc ADEGc FIYc
Plma35 → NP AGKTc EHNQSTYc LMc ELQVYc IVc ND WYc IVc KRc ALSTVc
Plma36 → WYc
Plma37 → EKQVc EKc
Plma38 → NW NV NL ND NW
Plma39 → ATc
Plma4 → ENRc ELc AIVc
Plma40 → DESc KRc NW DNc FYc DEc LTVc
Plma41 → GNc CGNc
Plma42 → HNSc
Plma43 → NN
Plma44 → ALVc NR QTVc EGc DGNc DGc CHKNVc NL IVc NI ESc NA RVc HKRc NE
Plma45 → NG KSc FVc NE MVc NR NA KRc ILc NP KRTc NG DKc HWc
Plma46 → FIMWYc DGc AFIMPSc AGQRSTc ADEKNQTc
Plma47 → DNc
Plma48 → QSc LVc NY
Plma49 → ENc NP
Plma5 → FWc
Plma50 → DSc NTc NA DLc NA AGc NV IVc NW NQc NE ND NF NN
Plma51 → NG
Plma52 → STc
Plma53 → NSc ENc
Plma54 → NPc MVYc
Plma55 → DEQc GNc CRc AEQSc
Plma56 → FYc NT NS AGc NR ILc HINQVc STc AKQRc DGc NK
Plma57 → FYc ADSc NF
Plma58 → KRc
Plma59 → NY NG RTc ILc DEc NA LRc NI KRc
Plma6 → AEQSc DEc DEQc NF DNSc DGKQRSc DNTc DESTc ILPVc DNc EPQRTWc AGKLQSc IKNYc NW
Plma60 → LMc PVc
Plma61 → EKNc LTc ARWc MNTc NG
Plma62 → NG
Plma63 → NA NE
Plma64 → AMc NG HPVc NA DETc AGc IMc NA
Plma65 → NA NG NTc NV
Plma66 → NRc KNRc EKYc IVc ELc NG NT FIc NH NW NSc AHNc GNQc NG
Plma67 → NQ NH NA NQ NY NG
Plma68 → DGc IQc EQc NY NM NV NF ND ITc RTc NP NL NP
Plma69 → AVc
Plma7 → NTc
Plma70 → NP NN NI NH ND APc GNc AGc IVc NT NA NP NL
Plma71 → STc NQ
Plma72 → NR
Plma73 → NL GPWc
Plma74 → NQTc GKc NW NN FHYc EQc STVc ESTc LMNc GYc NG GVc GWYc NSc NW
Plma75 → NC NT NN PSc DQc RWc NY NG NC AHc NR ETc NG
Plma76 → FYc
Plma77 → NN ATc DQc ALc IVc DNc AGc NSc
Plma78 → DSc KRc
Plma79 → FLYc FIVc NSWc
Plma8 → IVYc NE
Plma80 → AGSc ATc CLMYc NH NW NG NP
Plma81 → ARc RWc NN
Plma82 → GSc GPc NK NP NW NSc NN NTc NS NP NT AEc PSc KTc ND NF NW DKc GHc NR DSc NQc NW NL NP STc NW KQc
Plma83 → ND NQc NK KTc NY DNc NY STc GSc NG WYc ILc DEc NT
Plma84 → NP NK LTc GQc FIc QRc ND NSc ISc FWc NP NA NF NW
Plma85 → HQYc ACIc NN
Plma86 → NT
Plma87 → EKSc NL DNQc
Plma88 → NY PSc ND KNc NN FNc GNc ILc NTc
Plma89 → LMVc NK NR MVc NL LMc ILc NL NV NT NG NL NF LMc NS NL CFSc AGTc FIVc ATc ASc
Plma9 → NG FGWc GVc NN AKNQSc NE NL EQc CHNYc NY
Plma90 → STc AVc NS NA FLQc ATc NG GNSc SVc
Plma91 → NF FWYc DEc PSc FLc DNSc AGNSYc FYc DNc DRSTc EGSc FLRTc NW EQSc
Plma92 → EKc AKMc ADGSc DGNVc
Plma93 → WYc APSTc NN NG DEGKNQSc MPVc NF DNc ACc RTVWc NW
Plma94 → KRYc APc ENSc NQc NV
Plma95 → NSTc FMc STc
Plma96 → NSc DLc NG EKc LMc KRc NL AGSc NL NT NS PSc ASc NYc
Plma97 → NSc GKc
Plma98 → FYc DKSTc ACGSc AGc NE FLNYc NR STc KNTVc DNQSc FINTYc FYc

Plma99 → GHc

PlmaEx1 → NM NF NK NK NW NK NK NF NG NI NS NS NL NA NL NV NL NV NA NA NV NA NF NT NG NW NS NA NK NA NS NA NA ND NA NS NQ NI NV NS NE NM NG NA NG NW NN NL NG NN NQ NL NE NA NA NV NN NG NT NP NN NE NT NA NW NG NN NP NT NV NT NP NE NL NI NK NK NV NK NA NA NG NF NK NS NI NR NI NP NV NS NY NL NN NN NI NG NS NA NP NN NY NT NI NN NA NA NW NL NN NR NI NQ NQ NV NV ND NY NA NY NN NE NG NL NY NV NI NI NN NI NH NG ND NG NY NN NS NV NQ NG NG NW NL NL NV NN NG NG NN NQ NT NA NI NK NE NK NY NK NK NV NW NQ NQ NI NA NT NK NF NS NN NY NN ND NR NL NI NF NE NS NM NN NE NV NF ND NG NN NY NG NN NP NN NS NA NY NY NT NN NL NN NA NY NN NQ NI NF NV ND NT NV NR NQ NT NG NG NN NN NN NA NR NW NL NL NV NP NG NW NN NT NN NI ND NY NT NV NG NN NY NG NF NT NL NP NT ND NN NY NR NS NS NA NI NP NS NS NQ NK NR NI NM NI NS NA NH NY NY NS NP NW ND NF NA NG NE NE NN NG NN NI NT NQ NW NG NA NT NS NT NN NP NA NK NK NS NT NW NG NQ NE ND NY NL NE NS NQ NF NK NS NM NY ND NK NF NV NT NQ NG NY NP NV NV NI NG NE NF NG NS NI ND NK NT NS NY ND NS NS NN NN NV NY NR NA NA NY NA NK NA NV NT NA NK NA NK NK NY NK NM NV NP NV NY NW ND NN NG NH NN NG NQ NH NG NF NA NL NF NN NR NS NN NN NT NV NT NQ NQ NN NI NI NN NA NI NM NQ NG NM NQ

PlmaEx10 → NA NR NG NN NT NL NF NY NL NS NL NL NL NG NV NF NH NQ NC NR NS NQ NC NA NT NP NS NP NT NT NA NS NG NT NH NA NP NT NG NE NI NC NS NG ND NL NI NF NE ND NE NF ND NE NL ND NM

PlmaEx100 → NL NI NK NW NG NG NN NA NT NR NL NA NN NP NE

PlmaEx101 → NN NS NA NP NG NE NG NQ

PlmaEx102 → NL NT NL NT NV NR NR NL NR NQ ND ND NT NK NA NS NR NR ND NQ NS NG NN NP NT NP

PlmaEx103 → NA

PlmaEx104 → NL NR NA NN NE NS NH NP NA

PlmaEx105 → NE NG NR NM NA NT NT NP NH NS NG

PlmaEx106 → ND NN NK NS NH NH NT NI NH NS NK NW NG NN NT NL NG NH NA NN ND NP NK NK NT NH NA NQ NA NN NV NP NA ND NQ

PlmaEx11 → NM NI NA NA NV NL NF NA NL NV NA NS NA NL NA NA ND NI NP NK NP NR NL NT NH NL NA NS NG NE NV NE NL NS NL NE NG NS NP NG NM NK NV NF ND NV NK NY NS NV NK NG NQ NK NH NQ NG NP NF NH NQ NG NA ND NG NR NW NY NH NR NN NH NG NT ND NY NN NG NK NE NK NI NK NY NT NI NT NA NE NI ND NG NK NT NV NE NS NK NG NR NI NH NP NE NE NR NS NV NA NL NV NP NP NK NI NV NK NR NC NS NS NV NF NR ND ND NF NN NG NA NF NN NP

PlmaEx12 → ND NR

PlmaEx13 → NM NV NK NS NK NY NL NV NF NI NS NV NF NS NL NL NF NG NV NF NV NV NG NF NS NH NQ NG NV NK NA NE NE NE NR NP NM NG NT NA

PlmaEx14 → NM NK NS NI NI NS NI NA NA NL NS NV NL NG NL NI NS NK NT NM NA NA NP NA NP NA NP NV NP NG NT NA NW NN NG NS NH ND NV NM ND NF NN NY NH

PlmaEx15 → NM NP NL NG NK NG NA NA NT NS NE NT NI NS NK ND NS NS NF NP NI NE NK NR NM NQ NS NP NS NL NK NF NA NM NT NL NQ NN NL NT NL NL NV NF NS NA NL NS NL NN NA NQ

PlmaEx16 → NM NN NR NP NT NL NP NT NF NL NA NA NA NL NL NG NV NS NA NL NA NA NE NP NG NL

PlmaEx17 → NM NF NR NT NL NT NL NL NL NV NI NF NG NA NP NS NF NA NW NS NQ NS NN NE NN NS NK NP NL

PlmaEx18 → NM NT NM NA

PlmaEx19 → NK NV NG ND NN NL NV NR NT NS NQ NQ NF NQ NI NY NT NK ND NN NV NK NI NK NG ND NN NL NV NL NS NA NQ NY NN NS NT NI NY

PlmaEx2 → NM NK NK NI NF NT NI NL NL NL NL NF NI NA NN NT NS NL NA NF NK NT NT NT NA NE NE NW

PlmaEx20 → NH NR NL NK NP NW NG NE NR NA NW NR NA NE NN NV NW NQ NE NN NG NI NL NS NI NQ NA NK NY NE NP NH NT ND NT NK NG NN NE NY NF

PlmaEx21 → NT NG NY NY NL NN NN ND NP NA NT NW

PlmaEx22 → NT NG NV NA NA NN NQ NE NL NQ NY NY NT ND

PlmaEx23 → NC NE NR NG NV NS NV NG NA NE NP NE NS NG NR NN NC NL NI NL NT NA NT NR NE NN NY

PlmaEx24 → NR NA NR NI NE NN

PlmaEx25 → NS NT NQ NN NV NY NV NQ ND NG NK NL NN NI NK NA NM NN ND NS NK NS NF NP NQ ND NP NN NR NY NA NQ NY NS NS NG NK NI NN NT NK ND NK NL NS NL NK

PlmaEx26 → NG NS NA ND NG NK NR NV NT NE NI NA ND NN NR NL NY NI NN NC NF NK NG NS NN NG NK NI NY NS NG NR NV NY NA NH NE NT NQ NG NW NL

PlmaEx27 → NN NR NG NA

PlmaEx28 → NT NV NS NS NM NF NL NY NQ NN NG NS NE NI NA

PlmaEx29 → NC NI ND NN NE NG NR NS NY NH NT NV NH NS NN NW NT NY ND NL NN NQ NK NN NN NP NR NS NS NF NS NV NP NV ND NF ND

PlmaEx3 → NM NN NV NP NI NR NS NA NT NA NA NA NT NA NT NR NA NL NA NL NA NL NV NL NS NL NA NG NP NA NA NA NA NP NG NA NP NA NS NG NP ND NA NP NA NG NH NR NW NV NL

PlmaEx30 → NM ND NS NW NN ND ND NW NT NP

PlmaEx31 → NT NH NA NL NH NT NQ NN NA NN NG NN NN NG NR NN NW NN NA NS NT NA NL NA NE NA NE NG

PlmaEx32 → NF NG NT NL NH NG NP NG NY NS NG NG NS NS NY NG NG NQ NL NL NA NG NA NP NW NY NQ

PlmaEx33 → NG NQ NW NP NV NN NQ NS NS NG NG ND NY NH NF NP NE NG NQ NT NF NA NN

PlmaEx34 → NL ND NE ND NP NN NR NE NP NY NF NR NM NT NI NT NR NT NQ NA NT NR NA NT NN NC NT NW NA NT NT NS NT NS NP NT

PlmaEx35 → NG NR NN NM NM NN NI NP NI NS NQ NS NF NW NQ NK NG NG NF NG NG NN NN NI NW NG NS

PlmaEx36 → ND NA NL NL NG NT NF NA NP NP ND NG NG NF NW NE NW NG ND NL ND NS NS NG NF NA NN NP NW NR NT NS NK

PlmaEx37 → NG NK NE NL NL NN NF NN NN ND NG NK NN

PlmaEx38 → NI NK NY NR NT NT NY NE NN NQ NM NF NC NK NG NC NK NG NI NV NK NL NP NT NH NP NQ NF NI NI NF NN NT NA NV ND NH NF NA NP NP

PlmaEx39 → NV NA NN NY NS NF NI NI NS ND NP NS NN NN NK NS NV NG NC NP NI NV NI NP NN NR ND NF NY NL NI NL NN NF NA NV NI NE NK NY NA NK NP NE NH NY NK NE

PlmaEx4 → NM NK NN NI NK NN NC NL NT NY NI NS NI NC NA NF NL NF NV NI NA NS NA NS NH NK NS NT NG NS NP NS NQ NH NP ND NV NI NN NY NN ND NF NT NL NQ

PlmaEx40 → NV NY NR NT NM ND NV NS NN ND NS NF NS NE NV NR NK NE NY NF NV NI NF NN NM NA NI NG NG NQ NW NP NG NY ND NI

PlmaEx41 → NS NN NA NT NG NY NT NG NN NI NA NP NN ND

PlmaEx42 → NA NW NG NG NQ NQ NG NV

PlmaEx43 → NR NW NV NE

PlmaEx44 → NQ

PlmaEx45 → NK NG NS NK NS NS NF

PlmaEx46 → NV NE NE NT NT NT NK NY NT NG NP NM NI NT NV NT NE ND NA NV NE NT NC NS NG NK NW NG NS NY NF NG NS ND NW NA NG NA NS NG NT NS NK NP NT ND NK NA NV NT NG NA NT NM NN NI NT NS NI NG

NN NS NQ NW NG NV NQ NQ NY NL NK NG NL NH NY NY NP NG NR NT NY NN NY NS NF NT NM NT NS ND NV ND NK NR NV NF NV NK NV NA NG ND NG ND NE NQ NI NF NG NE NY NI ND NL NK NA NG NV NP NY NN NF NS NR NQ NV NT NL NA NK ND
PlmaEx47 → NY NR NF NP NN NE NR NL NT ND NP NE NA ND NW NR NH NW
PlmaEx48 → NG NT NG NN NV NA NM ND NG NQ NG NH NL NV NI NT NA NR NK NG NS NG NG NH NN ND NC NW NN NG NT NC NQ
PlmaEx49 → NF NT NQ NQ
PlmaEx5 → NM NH NF NL NR NS NT NL NR NG NL NI NL NC NA NL NL NL NI NG NS NS NS NL NL NQ NA NV NN NF NR NE NS ND NR ND NK NP NQ NY NL NK ND NI NK NN NR NT NV NQ NA NQ NI NL NY NV NG NE NH NW NI NY NL NK NR NN NS NT NT NA NR NR NL NP NL NG NI NL NS NQ NQ NS NI NE NH NA NQ NA NW NA ND NR NF NQ NS NK NL NS NE NT NH NQ NR NE NL NL NA NI NA NA NT NA NS NP NG NQ NP NL NI NL ND NP ND NP NS NK NQ NW NV NL NK NE ND
PlmaEx50 → NT
PlmaEx51 → NG
PlmaEx52 → NT NG
PlmaEx53 → NN NP NT
PlmaEx54 → NT NN NY NV NA NP NN ND NL NT NT
PlmaEx55 → NN NH NN NH NG NV NN NE NV NG
PlmaEx56 → ND NA NG NQ NN NK NF NY NL NT NH NG ND NV NS NG ND NW NS NH
PlmaEx57 → NI NN NM NY NM NM NT NH NV NE NS NS NN NP NA NG NF ND NA
PlmaEx58 → NY NK NR NN NA NV NI NN ND NG NC NL NQ NL NS NI ND NQ NK NW NT NN ND NK NN NP ND NW ND NP NR
PlmaEx59 → NT NP NN NN ND NK NF NE
PlmaEx6 → NM NK NK NL NL NI NT NT NA NA NL NA NS NL NW NL NL NP NT NA NC NG NN ND ND ND NG NT NT NP NP NP NT ND NP ND NS NP NE NA NP NI NE NG NY NS NL NY
PlmaEx60 → NT NA ND NG NL NM NR NL NT NI NA NK NK NT NT NS
PlmaEx61 → NH NY NG NL NF NE NV NS NM NK NP NA NK NV NE NG NT NV NS NS NF NF NT
PlmaEx62 → ND NN NT NP NV NY NS NY NY ND NW NV NR NY NT NP NL NQ NN NY NQ NI NH NQ
PlmaEx63 → NR NK NK
PlmaEx64 → ND NA NP NR NN
PlmaEx65 → NV NQ NH NS NS NQ NG NQ NN NP NW NG
PlmaEx66 → NV NR NK NT NE NG NG NQ NV NS NN NL NT NG NT NQ NG NL NR NF NN NL NW NS NS NE NS NA NA NW NV NG NQ NF ND NE NS NK NL NP NL NF NQ NF NI NN NW NV NK NV NY NK NY NT NP NG NQ NG NE NG NG NS ND NF NT NL ND NW NT ND NN NF ND NT NF ND NG NS NR NW NG NK NG ND NW NT NF ND NG NN NR NV ND NL NT ND NK NN NI NY NS NR ND NG NM NL NI NL NA NL NT NR NK NG NQ NE NS NF NN NG NQ NV NP NR ND ND NE NP NA NP NQ NS NS NS NS NA NP NA NS NS NS NS NV NP NA NS NS NS NS NV NP NA NS NS NS NS NA NF NV NP NP NS NS NS NS NA NT NN NA NI NH NG NM NR NT NT NP NA NV NA NK NE NH NR NN NL NV NN NA NK NG NA NK NV NN NP NN NG NH NK NR NY NR NV NN NF NE NH
PlmaEx67 → NS
PlmaEx68 → NI NT NK NE NR NL NT NT
PlmaEx69 → ND NE NR NF NT NL NG NF ND NG NG ND
PlmaEx7 → NM NN NI NK NK NT NA NV NK NS NA NL NA NV NA NA NA NA NA NA NL NT NT NN NV NS NA NK ND NF NS NG NA NE NL NY NT NL NE NE NV NQ
PlmaEx70 → NP NS NH NS ND ND NF NN NY NT NG NK NP NQ NT NF NR NG
PlmaEx71 → NL NV NQ NN NG NH NL NE NL NL NA NS NR NK
PlmaEx72 → NR NV NH ND NG NN NL NE NL NL NA NQ NA NV NP NG NP NK NG NK NN
PlmaEx73 → NT NI NT NG NG NK NL NV NL NS NA NS NR NK NA NG NT ND NL
PlmaEx74 → NV NA ND NG NN NL NI NV NE NG NR NR NA NP ND NG NR NV NY NC NG
PlmaEx75 → NE NS NL NH NG
PlmaEx76 → NT ND NW NF NR NR NH NP NA NH NN NA NH NF NF NK NR NG NG NN NG ND NI NN NR NQ NG NH NH NT NT ND NR NR NW ND NN NQ NY NH NR NY
PlmaEx77 → NR NG NG NA NG NG NA NG NG NS NA ND NI NG NG NN NR NG ND NG NE NP NG NI NA NG NS NN NY NH NM NF NE NR ND NP NV NS NN ND NV NI NK ND NH NG NG ND NG NH NH NH NP NP NS ND NG NV NA NY NR ND NG NF NH NT NF
PlmaEx78 → NL NA NR NS NQ NK NG NY NF NA ND NG NS NY NG NY NN NG NE NT NG NQ NV NF NG ND NG NA
PlmaEx79 → NR NL NV NR NR NL NN NR NS NN ND NL NR ND NP NR NS NR NF NF
PlmaEx8 → NM NC NT NM NP NL NM NK NL NK NK NM NM NR NR NT NA NF NL NL NS NV NL NI NG NC NS NM NL NG NS ND NR NS ND NK NA NP NH NW NE NL NV NW NS ND NE NF ND NY NS NG NL NP ND NP
PlmaEx80 → ND NL NV NR NQ NM NW NL NP NG NQ NI NP ND NP NT NG NQ NY NL
PlmaEx81 → NP NS ND NA
PlmaEx82 → NP NT ND NA
PlmaEx83 → NS NW NT NL NE NK NI NE NA ND
PlmaEx84 → NA NV
PlmaEx85 → NN NS NP NA NL NP
PlmaEx86 → NA NR NM
PlmaEx87 → NI NN NR NL NV NT
PlmaEx88 → ND NS NA NH NE NT NK NM NK NT NN NY NH NL NF NK NR NS NK ND NG NE NK NI NL NR ND NY NC NL NS NA NS NH NQ NL NP NN NQ NE NP NL NR NK NG NF NH
PlmaEx89 → NG NT NL NW NA ND ND NF NR
PlmaEx9 → NK NA NS NG NM NI NS NV NT NV ND NE NN NT NS NK NE NA NR NN NG NA NI NN NV NK NL NG NS NK NQ NV NS NI NS NV NT NQ NA NG NK NT NV NT NP NI NE NG NG NE NM NV NI NP NE NG NY NK NL NV NW NN ND NE NF NN NE NG NS NE NL NN NS NT ND NW NR NH NE NV NQ NK
PlmaEx90 → NL NA NG NA ND NI
PlmaEx91 → NK ND ND NF NK NE NS NI NA NA NG NF NT
PlmaEx92 → NA NA NP NP NN NG NQ NG NL NT NP NT ND
PlmaEx93 → NG NL NV NP NT ND
PlmaEx94 → NV NG NR NR NG NR
PlmaEx95 → NV NP NQ NA NV NQ NA NE NI NA NT NT ND NT NE NI NA NL NN NF NS NK NA NR NP NY NL NF NY NQ NV NL NQ NA NE NG NL ND NA NP NN NW NT NP NA NL NT ND NL NQ NF NN NS NQ NG NL NA NS NL NS NR NT NI NE NS NT NQ NH NF NF NR NL NQ NT ND NT ND NA NP NV NL NA NT NF ND NW NE NE NY NA NG NE NW NY NE NG NY NN NQ NT NE NT NH NN NG NI NS NL NL NA NS NG NR NS NA NV NH ND NG NG ND NG NR NR NG NS NN NG NQ NA NL NK NA NN NY NA NQ ND NA NV NP NG NA NY NQ NV NQ NL NA NG NA NI NK ND NF NQ NV NL NS NV ND NV NS NA ND NE NG NA NQ NA NG NS NV NY NL NE NG NR NL NN NG NA NL NQ NW NT NL ND NP NV NN NQ NS NA NL NQ NT NY NT NT NA NT NS NG NE NL NT NA NL NI ND NE NI NL NW NH NG NP NT NA NT

NQ NA NN NP NG NG NT NV NW NN NN NS NL ND NN NL NT NV NL NV NR NQ
PlmaEx96 → NN NG NK ND NL NN NP ND NY NP NH
PlmaEx97 → NI
PlmaEx98 → NG NG NY ND NV NK NE NK NR NK NT ND NE NH NR NI ND NM NN NL NH NT NR NE NM NI ND NG NV NE NT NW NR NR NP NQ NH NW NP NE NV NC NA NN NH NI ND
PlmaEx99 → NG NN NR NI NE NG NN NE NR NI NM ND NH NN NL NH NS NI NL NS NN NG NT NG NG NI NA NG NR NS NW NQ NR NP NN ND NA NR NF NK NA NT NQ NA NI NE NY NH

règles terminales (acides aminés)

AA → NA | NC | ND | NE | NF | NG | NH | NI | NK | NL | NM | NN | NP | NQ | NR | NS | NT | NV | NW | NY | 'X'
NA → 'A'
NC → 'C'
ND → 'D'
NE → 'E'
NF → 'F'
NG → 'G'
NH → 'H'
NI → 'I'
NK → 'K'
NL → 'L'
NM → 'M'
NN → 'N'
NP → 'P'
NQ → 'Q'
NR → 'R'
NS → 'S'
NT → 'T'
NV → 'V'
NW → 'W'
NY → 'Y'

règles introduites pour faciliter l'analyse des gaps

Gap → AA | Gap AA

GPlma107 → Plma107 | Plma107 Gap
GPlma115 → Plma115 | Plma115 Gap
GPlma121 → Plma121 | Plma121 Gap
GPlma122 → Plma122 | Plma122 Gap
GPlma124 → Plma124 | Plma124 Gap
GPlma129 → Plma129 | Plma129 Gap
GPlma153 → Plma153 | Plma153 Gap
GPlma18 → Plma18 | Plma18 Gap
GPlma193 → Plma193 | Plma193 Gap
GPlma199 → Plma199 | Plma199 Gap
GPlma20 → Plma20 | Plma20 Gap
GPlma206 → Plma206 | Plma206 Gap
GPlma209 → Plma209 | Plma209 Gap
GPlma210 → Plma210 | Plma210 Gap
GPlma211 → Plma211 | Plma211 Gap
GPlma212 → Plma212 | Plma212 Gap
GPlma213 → Plma213 | Plma213 Gap
GPlma214 → Plma214 | Plma214 Gap
GPlma215 → Plma215 | Plma215 Gap
GPlma216 → Plma216 | Plma216 Gap
GPlma217 → Plma217 | Plma217 Gap
GPlma218 → Plma218 | Plma218 Gap
GPlma219 → Plma219 | Plma219 Gap
GPlma220 → Plma220 | Plma220 Gap
GPlma221 → Plma221 | Plma221 Gap
GPlma226 → Plma226 | Plma226 Gap
GPlma24 → Plma24 | Plma24 Gap
GPlma29 → Plma29 | Plma29 Gap
GPlma3 → Plma3 | Plma3 Gap
GPlma32 → Plma32 | Plma32 Gap
GPlma38 → Plma38 | Plma38 Gap
GPlma44 → Plma44 | Plma44 Gap
GPlma47 → Plma47 | Plma47 Gap
GPlma48 → Plma48 | Plma48 Gap
GPlma50 → Plma50 | Plma50 Gap
GPlma63 → Plma63 | Plma63 Gap
GPlma64 → Plma64 | Plma64 Gap
GPlma66 → Plma66 | Plma66 Gap
GPlma67 → Plma67 | Plma67 Gap
GPlma7 → Plma7 | Plma7 Gap
GPlma74 → Plma74 | Plma74 Gap
GPlma75 → Plma75 | Plma75 Gap
GPlma77 → Plma77 | Plma77 Gap
GPlma8 → Plma8 | Plma8 Gap
GPlma81 → Plma81 | Plma81 Gap

GPlma83 → Plma83 | Plma83 Gap
GPlma84 → Plma84 | Plma84 Gap
GPlma85 → Plma85 | Plma85 Gap

GXf103X1 → Xf103X1 | Xf103X1 Gap
GXf103X33 → Xf103X33 | Xf103X33 Gap
GXf103X5 → Xf103X5 | Xf103X5 Gap
GXf39X13 → Xf39X13 | Xf39X13 Gap
GXf39X133 → Xf39X133 | Xf39X133 Gap
GXf39X24 → Xf39X24 | Xf39X24 Gap
GXf39X29 → Xf39X29 | Xf39X29 Gap
GXf39X32 → Xf39X32 | Xf39X32 Gap
GXf39X4 → Xf39X4 | Xf39X4 Gap
GXf39X50 → Xf39X50 | Xf39X50 Gap
GXf39X67 → Xf39X67 | Xf39X67 Gap
GXf39X76 → Xf39X76 | Xf39X76 Gap
GXf39X78 → Xf39X78 | Xf39X78 Gap
GXf39X8 → Xf39X8 | Xf39X8 Gap
GXf73X15 → Xf73X15 | Xf73X15 Gap
GXf73X18 → Xf73X18 | Xf73X18 Gap
GXf73X49 → Xf73X49 | Xf73X49 Gap
GXf83X31 → Xf83X31 | Xf83X31 Gap
GXfnew1X21 → Xfnew1X21 | Xfnew1X21 Gap
GXfnew1X26 → Xfnew1X26 | Xfnew1X26 Gap
GXfnew1X47 → Xfnew1X47 | Xfnew1X47 Gap
GXfnew1X58 → Xfnew1X58 | Xfnew1X58 Gap
GXfnew1X8 → Xfnew1X8 | Xfnew1X8 Gap

règles permettant l'analyse d'une position dans un bloc plma

ACFIVc → NA | NC | NF | NI | NV
ACGSc → NA | NC | NG | NS
ACISc → NA | NC | NI | NS
ACIc → NA | NC | NI
ACLSc → NA | NC | NL | NS
ACSc → NA | NC | NS
ACc → NA | NC
ADEGNQRSTc → NA | ND | NE | NG | NN | NQ | NR | NS | NT
ADEGRSc → NA | ND | NE | NG | NR | NS
ADEGc → NA | ND | NE | NG
ADEHKPQRVc → NA | ND | NE | NH | NK | NP | NQ | NR | NV
ADEHSYc → NA | ND | NE | NH | NS | NY
ADEKNQTc → NA | ND | NE | NK | NN | NQ | NT
ADENQRTc → NA | ND | NE | NN | NQ | NR | NT
ADERYc → NA | ND | NE | NR | NY
ADESTc → NA | ND | NE | NS | NT
ADEc → NA | ND | NE
ADFLNRc → NA | ND | NF | NL | NN | NR
ADGSc → NA | ND | NG | NS
ADHIKQRSTVc → NA | ND | NH | NI | NK | NQ | NR | NS | NT | NV
ADHIKRTWYc → NA | ND | NH | NI | NK | NR | NT | NW | NY
ADKQSTVc → NA | ND | NK | NQ | NS | NT | NV
ADPc → NA | ND | NP
ADQc → NA | ND | NQ
ADSc → NA | ND | NS
ADc → NA | ND
AEGMc → NA | NE | NG | NM
AEHSVYc → NA | NE | NH | NS | NV | NY
AEKLc → NA | NE | NK | NL
AEKPc → NA | NE | NK | NP
AELPc → NA | NE | NL | NP
AEPSWc → NA | NE | NP | NS | NW
AEQRSc → NA | NE | NQ | NR | NS
AEQSc → NA | NE | NQ | NS
AEc → NA | NE
AFILMTc → NA | NF | NI | NL | NM | NT
AFIMPSc → NA | NF | NI | NM | NP | NS
AFLc → NA | NF | NL
AFYc → NA | NF | NY
AFc → NA | NF
AGHQSYc → NA | NG | NH | NQ | NS | NY
AGKLQSc → NA | NG | NK | NL | NQ | NS
AGKQRSTc → NA | NG | NK | NQ | NR | NS | NT
AGKTc → NA | NG | NK | NT
AGMNQSc → NA | NG | NM | NN | NQ | NS
AGNPSc → NA | NG | NN | NP | NS
AGNSTVc → NA | NG | NN | NS | NT | NV
AGNSYc → NA | NG | NN | NS | NY
AGNc → NA | NG | NN
AGQRSTc → NA | NG | NQ | NR | NS | NT
AGSc → NA | NG | NS
AGTc → NA | NG | NT
AGc → NA | NG

AHIVYc → NA | NH | NI | NV | NY
AHNc → NA | NH | NN
AHc → NA | NH
AIKLTVc → NA | NI | NK | NL | NT | NV
AIMPc → NA | NI | NM | NP
AIMVc → NA | NI | NM | NV
AIMc → NA | NI | NM
AIVc → NA | NI | NV
AIc → NA | NI
AKMc → NA | NK | NM
AKNQSc → NA | NK | NN | NQ | NS
AKNSTc → NA | NK | NN | NS | NT
AKQRc → NA | NK | NQ | NR
AKQVc → NA | NK | NQ | NV
AKc → NA | NK
ALMPVc → NA | NL | NM | NP | NV
ALSTVc → NA | NL | NS | NT | NV
ALVc → NA | NL | NV
ALc → NA | NL
AMc → NA | NM
ANc → NA | NN
APRSVc → NA | NP | NR | NS | NV
APSTc → NA | NP | NS | NT
APTc → NA | NP | NT
APc → NA | NP
AQc → NA | NQ
ARSc → NA | NR | NS
ARVc → NA | NR | NV
ARWc → NA | NR | NW
ARc → NA | NR
ASTWc → NA | NS | NT | NW
ASTc → NA | NS | NT
ASWc → NA | NS | NW
ASc → NA | NS
ATc → NA | NT
AVc → NA | NV
CDKQRc → NC | ND | NK | NQ | NR
CDNQSYc → NC | ND | NN | NQ | NS | NY
CFILMVc → NC | NF | NI | NL | NM | NV
CFILVc → NC | NF | NI | NL | NV
CFSc → NC | NF | NS
CFc → NC | NF
CGNc → NC | NG | NN
CHKNVc → NC | NH | NK | NN | NV
CHNYc → NC | NH | NN | NY
CIc → NC | NI
CLMYc → NC | NL | NM | NY
CLVWc → NC | NL | NV | NW
CRc → NC | NR
CSc → NC | NS
CWc → NC | NW
DEFILMNTVc → ND | NE | NF | NI | NL | NM | NN | NT | NV
DEFLNVc → ND | NE | NF | NL | NN | NV
DEGHKNSTc → ND | NE | NG | NH | NK | NN | NS | NT
DEGKNQSc → ND | NE | NG | NK | NN | NQ | NS
DEGNRc → ND | NE | NG | NN | NR
DEGSc → ND | NE | NG | NS
DEHKLNQSTYc → ND | NE | NH | NK | NL | NN | NQ | NS | NT | NY
DEHNPRc → ND | NE | NH | NN | NP | NR
DEHQTVc → ND | NE | NH | NQ | NT | NV
DEHQc → ND | NE | NH | NQ
DEHc → ND | NE | NH
DEKPc → ND | NE | NK | NP
DENQRc → ND | NE | NN | NQ | NR
DENRSTc → ND | NE | NN | NR | NS | NT
DENRc → ND | NE | NN | NR
DENc → ND | NE | NN
DEQSc → ND | NE | NQ | NS
DEQc → ND | NE | NQ
DERc → ND | NE | NR
DESTc → ND | NE | NS | NT
DESc → ND | NE | NS
DETc → ND | NE | NT
DEc → ND | NE
DFGc → ND | NF | NG
DGHKNc → ND | NG | NH | NK | NN
DGKQRSc → ND | NG | NK | NQ | NR | NS
DGNPRSTYc → ND | NG | NN | NP | NR | NS | NT | NY
DGNSTc → ND | NG | NN | NS | NT
DGNVc → ND | NG | NN | NV
DGNc → ND | NG | NN
DGc → ND | NG
DHNc → ND | NH | NN
DILNTVc → ND | NI | NL | NN | NT | NV

DKNSTc → ND | NK | NN | NS | NT
DKPQRSTVc → ND | NK | NP | NQ | NR | NS | NT | NV
DKQc → ND | NK | NQ
DKSTc → ND | NK | NS | NT
DKTc → ND | NK | NT
DKc → ND | NK
DLNRTVc → ND | NL | NN | NR | NT | NV
DLNRTc → ND | NL | NN | NR | NT
DLc → ND | NL
DNPSc → ND | NN | NP | NS
DNQSc → ND | NN | NQ | NS
DNQc → ND | NN | NQ
DNRTc → ND | NN | NR | NT
DNRWc → ND | NN | NR | NW
DNRc → ND | NN | NR
DNSYc → ND | NN | NS | NY
DNSc → ND | NN | NS
DNTc → ND | NN | NT
DNc → ND | NN
DPQTc → ND | NP | NQ | NT
DPc → ND | NP
DQTc → ND | NQ | NT
DQc → ND | NQ
DRSTc → ND | NR | NS | NT
DRYc → ND | NR | NY
DRc → ND | NR
DSc → ND | NS
DTc → ND | NT
DVc → ND | NV
DYc → ND | NY
EGKNPc → NE | NG | NK | NN | NP
EGSc → NE | NG | NS
EGc → NE | NG
EHKQc → NE | NH | NK | NQ
EHNQSTYc → NE | NH | NN | NQ | NS | NT | NY
EHNYc → NE | NH | NN | NY
EHSYc → NE | NH | NS | NY
EIKQc → NE | NI | NK | NQ
EIVc → NE | NI | NV
EKNc → NE | NK | NN
EKQRSc → NE | NK | NQ | NR | NS
EKQRTVc → NE | NK | NQ | NR | NT | NV
EKQVc → NE | NK | NQ | NV
EKQc → NE | NK | NQ
EKRSc → NE | NK | NR | NS
EKSc → NE | NK | NS
EKYc → NE | NK | NY
EKc → NE | NK
ELQVYc → NE | NL | NQ | NV | NY
ELc → NE | NL
ENQRTc → NE | NN | NQ | NR | NT
ENRc → NE | NN | NR
ENSc → NE | NN | NS
ENc → NE | NN
EPQRTWc → NE | NP | NQ | NR | NT | NW
EPQc → NE | NP | NQ
EPc → NE | NP
EQSc → NE | NQ | NS
EQc → NE | NQ
ERSTc → NE | NR | NS | NT
ERYc → NE | NR | NY
ERc → NE | NR
ESTc → NE | NS | NT
ESc → NE | NS
ETVc → NE | NT | NV
ETc → NE | NT
EVc → NE | NV
FGTc → NF | NG | NT
FGWc → NF | NG | NW
FGc → NF | NG
FHKQYc → NF | NH | NK | NQ | NY
FHKYc → NF | NH | NK | NY
FHYc → NF | NH | NY
FHc → NF | NH
FILRVYc → NF | NI | NL | NR | NV | NY
FILVYc → NF | NI | NL | NV | NY
FILVc → NF | NI | NL | NV
FILc → NF | NI | NL
FIMWYc → NF | NI | NM | NW | NY
FINTYc → NF | NI | NN | NT | NY
FIVYc → NF | NI | NV | NY
FIVc → NF | NI | NV
FIYc → NF | NI | NY
FIc → NF | NI

FKQRVc → NF | NK | NQ | NR | NV
FLNYc → NF | NL | NN | NY
FLQc → NF | NL | NQ
FLRTc → NF | NL | NR | NT
FLSTYc → NF | NL | NS | NT | NY
FLYc → NF | NL | NY
FLc → NF | NL
FMTVWc → NF | NM | NT | NV | NW
FMWYc → NF | NM | NW | NY
FMc → NF | NM
FNYc → NF | NN | NY
FNc → NF | NN
FRTWYc → NF | NR | NT | NW | NY
FTc → NF | NT
FVc → NF | NV
FWYc → NF | NW | NY
FWc → NF | NW
FYc → NF | NY
GHc → NG | NH
GKNRc → NG | NK | NN | NR
GKRc → NG | NK | NR
GKc → NG | NK
GNQc → NG | NN | NQ
GNSTc → NG | NN | NS | NT
GNSc → NG | NN | NS
GNTc → NG | NN | NT
GNc → NG | NN
GPSc → NG | NP | NS
GPWc → NG | NP | NW
GPc → NG | NP
GYc → NG | NY
GQc → NG | NQ
GRc → NG | NR
GSTc → NG | NS | NT
GSc → NG | NS
GTc → NG | NT
GVc → NG | NV
GWYc → NG | NW | NY
GWc → NG | NW
HIKLMRYc → NH | NI | NK | NL | NM | NR | NY
HINQVc → NH | NI | NN | NQ | NV
HKLTc → NH | NK | NL | NT
HKNRc → NH | NK | NN | NR
HKRc → NH | NK | NR
HKTc → NH | NK | NT
HLRc → NH | NL | NR
HMc → NH | NM
HNSc → NH | NN | NS
HNc → NH | NN
HPVc → NH | NP | NV
HPc → NH | NP
HQYc → NH | NQ | NY
HQc → NH | NQ
HRc → NH | NR
HWc → NH | NW
HYc → NH | NY
IKLQRc → NI | NK | NL | NQ | NR
IKMPc → NI | NK | NM | NP
IKNRTc → NI | NK | NN | NR | NT
IKNYc → NI | NK | NN | NY
IKQRc → NI | NK | NQ | NR
IKc → NI | NK
ILMTVc → NI | NL | NM | NT | NV
ILMVc → NI | NL | NM | NV
ILMc → NI | NL | NM
ILPQSc → NI | NL | NP | NQ | NS
ILPVc → NI | NL | NP | NV
ILQVc → NI | NL | NQ | NV
ILTc → NI | NL | NT
ILVc → NI | NL | NV
ILc → NI | NL
IMVc → NI | NM | NV
IMc → NI | NM
IPVc → NI | NP | NV
IPc → NI | NP
IQVc → NI | NQ | NV
IQc → NI | NQ
IRTVc → NI | NR | NT | NV
ISTc → NI | NS | NT
ISc → NI | NS
ITVc → NI | NT | NV
ITc → NI | NT
IVYc → NI | NV | NY
IVc → NI | NV

IYc → NI | NY
KLNYc → NK | NL | NN | NY
KMQRYc → NK | NM | NQ | NR | NY
KMc → NK | NM
KNRc → NK | NN | NR
KNTVc → NK | NN | NT | NV
KNYc → NK | NN | NY
KNc → NK | NN
KPQVc → NK | NP | NQ | NV
KPVc → NK | NP | NV
KPc → NK | NP
KQRWc → NK | NQ | NR | NW
KQRc → NK | NQ | NR
KQSc → NK | NQ | NS
KQc → NK | NQ
KRSTc → NK | NR | NS | NT
KRTc → NK | NR | NT
KRYc → NK | NR | NY
KRc → NK | NR
KSc → NK | NS
KTVc → NK | NT | NV
KTc → NK | NT
KVc → NK | NV
KYc → NK | NY
LMNc → NL | NM | NN
LMQc → NL | NM | NQ
LMTVc → NL | NM | NT | NV
LMVc → NL | NM | NV
LMc → NL | NM
LNTVc → NL | NN | NT | NV
LPc → NL | NP
LQSTc → NL | NQ | NS | NT
LRc → NL | NR
LSc → NL | NS
LTVc → NL | NT | NV
LTc → NL | NT
LVc → NL | NV
LYc → NL | NY
MNTc → NM | NN | NT
MPVc → NM | NP | NV
MPc → NM | NP
MRSc → NM | NR | NS
MSc → NM | NS
MTc → NM | NT
MVYc → NM | NV | NY
MVc → NM | NV
NPc → NN | NP
NTc → NN | NT
NQSTVc → NN | NQ | NS | NT | NV
NQSc → NN | NQ | NS
NQTc → NN | NQ | NT
NQc → NN | NQ
NRSc → NN | NR | NS
NRc → NN | NR
NSTc → NN | NS | NT
NSWc → NN | NS | NW
NSc → NN | NS
NVWc → NN | NV | NW
NVc → NN | NV
NYc → NN | NY
PQc → NP | NQ
PSc → NP | NS
PTc → NP | NT
PVc → NP | NV
PYc → NP | NY
QRSc → NQ | NR | NS
QRTVc → NQ | NR | NT | NV
QRWYc → NQ | NR | NW | NY
QRc → NQ | NR
QSTYc → NQ | NS | NT | NY
QSc → NQ | NS
QTVc → NQ | NT | NV
QTc → NQ | NT
RSc → NR | NS
RTVWc → NR | NT | NV | NW
RTc → NR | NT
RVc → NR | NV
RWc → NR | NW
RYc → NR | NY
STVc → NS | NT | NV
STc → NS | NT
SVc → NS | NV
SWc → NS | NW
SYc → NS | NY

$\mathrm{TVc} \to \mathrm{NT} \mid \mathrm{NV}$
$\mathrm{TYc} \to \mathrm{NT} \mid \mathrm{NY}$
$\mathrm{VYc} \to \mathrm{NV} \mid \mathrm{NY}$
$\mathrm{WYc} \to \mathrm{NW} \mid \mathrm{NY}$

Table des figures

Résumé

Cette thèse propose une nouvelle approche de découverte de signatures de familles (et superfamilles) d'enzymes. Dans un premier temps, étant donné un échantillon aligné de séquences appartenant à une même famille, cette approche infère des grammaires algébriques caractérisant cette famille. Pour ce faire, de nouveaux principes de généralisation et de nouvelles classes de langages ont été introduites sur la base de la substituabilité locale. Un algorithme a également été développé à cet effet qui produit une grammaire réduite, conservant la structuration des exemples, d'un langage substituable. Dans un second temps, ce manuscrit présente une méthode de classification des séquences d'une superfamille en familles à l'aide d'une analyse de concepts formels basée sur l'alignement des séquences qui permet la détection de nouvelles familles et la découverte des motifs fonctionnels pour améliorer les signatures précédentes.

Mots clés

bioinformatique, enzyme, famille, inférence grammaticale, grammaire algébrique, substituabilité, analyse de concepts formels

Abstract

This thesis proposes a new approach to discover signatures of families (and superfamilies) enzymes. At first, given a sample of aligned sequences belonging to the same family, this approach infers context-free grammars characteristic of this family. To do this, new principles of generalization and new classes have been introduced based on substitutability. An algorithm has also been developed for this purpose, which produces a reduced grammar able to retain the structure of examples. In a second step, this manuscript presents a method for classification of a superfamily sequences into families with a formal concept analysis based on alignement sequences allowing detection of new families and the discovery of patterns to improve functional previous signatures.

Keywords

bioinformatics, enzyme, family, grammatical inference, context-free grammar, substitutability, formal concept analysis